Towards optimisation of digital filters and multirate filter banks through genetic algorithms

A Thesis Submitted to the University of Wales for the Degree of

DOCTOR of PHILOSOPHY

by

Gurvinder Singh Baicher

School of Computing and Engineering

University of Wales College, Newport

July, 2003

Contents

Chapter 3: Finite word length optimisation of IIR filters

Chapter 4: Optimisation and real-time implementation of a class of multirate quadrature mirror filter bank

Chapter 5: Optimisation of a class of M-channel uniform filter bank

Chapter 6: Optimisation of a class of M-channel non-uniform filter bank

Appendices

Appendix A

Appendix B

Appendix C

Appendix D

Appendix E

Appendix F

Declaration / Statements

DECLARATION

This work has not previously been accepted in substance for any degree and is not being concurrently submitted in candidature for any degree.

Signed ..... (candidate)
Date ..... 15 December 2003

STATEMENT 1

This thesis is the results of my own investigations, except where otherwise stated.

Other sources are acknowledged by footnotes giving explicit references. A bibliography is appended.

Signed ..... (candidate)
Date ..... 15 December 2003

STATEMENT 2

I hereby give consent for my thesis, if accepted, to be available for photocopying and for inter-library loan, and for the title and summary to be made available to outside organisations.

Signed ..... (candidate)
Date ..... 15 December 2003

Acknowledgements

To Dr Brian Turton and Professor Hefin Rowlands, for their advice, guidance and support at every stage of this effort.

To my wife Neeta, for her forbearance during the long and tedious hours involved with this work and being always understanding and supportive. Also to my children Neena and Vikram who have played their part in this effort.

Summary

This thesis is concerned with the issues of design and optimisation of digital filters and multirate filter banks. The main focus and contribution of this thesis is to apply the genetic algorithm (GA) technique and to draw some comparison with the standard gradient and non-gradient based optimisation methods. The finite word length (FWL) constraint affects the accuracy of a real-time digital filter frequency response. For the case of digital filters, this study is concerned with the optimisation of FWL coefficients using genetic algorithms. Some comparative study with the simple hill climber algorithms is also included. The outcome of this part of the study demonstrates a substantial improvement of the new results when compared with the simply rounded FWL coefficient frequency response.

The FWL coefficient optimisation process developed in the earlier Chapters is extended to the field of multirate filter banks. All multirate filter banks suffer from the problems of amplitude, phase and aliasing errors and, therefore, constraints for perfect reconstruction (PR) of the input signal can be extensive. The problem, in general, is reduced to relaxing constraints at the expense of errors and finding methods for minimising the errors. Optimisation techniques are thus commonly used for the design and implementation of multirate filter banks. In this part of the study, GAs have been used in two distinct stages. Firstly, for the design optimisation so that the overall errors are minimised and secondly for FWL coefficient optimisation of digital filters that form the sub-band filters of the filter bank. This process leads to an optimal realisation of the filter bank that can be applied to specific applications such as telephony speech signal coding and compression. One example of the optimised QMF bank was tested on a real-time DSP target system and the results are reported.

The multiple M-channel uniform and non-uniform filter banks have also been considered in this study for design optimisation. For a comparative study of the GA optimised results of the design stage of the filter bank, other standard methods such as the gradient based quasi-Newton and the non-gradient based downhill Simplex methods were also used. In general, the outcome of this part of study demonstrates that a hybrid approach of GA and standard method was the most efficient and effective process in generating the best results.

Nomenclature

Acronyms and abbreviations

1-D	one dimensional
2-D	two dimensional
A/D	analogue to digital
ADPCM	adaptive differential pulse code modulation
BIBO	bounded-input bounded-output
BP	band pass
COFF	common object file format
D/A	digital to analogue
DF	direct form
DFG	data flow graph
DSK	digital (signal processing) starter kit
DSP	digital signal processing
FFT	fast Fourier transform
FIR	finite impulse response
FWL	finite word length
GA	genetic algorithm
HP	high pass
IIR	infinite impulse response
IN	infinity norm
IP	integer programming
LP	low pass
LTI	linear time invariant
MELP	mixed excitation linear prediction
MOS	mean opinion score
MP	minimum phase
NMP	non minimal phase
NN	none norm
NUF	Non-uniform filter
PCM	pulse code modulation
PR	perfect reconstruction

QMF	quadrature mirror filter
ROC	region of convergence
SNR	signal to noise ratio
SOS	second order section
SP	selective pressure
SQNR	signal to quantisation noise ratio
VLSI	very large scale integration
WLS	weighted least square

List of Figures

List of Tables

Chapter 1 : Introduction and Overview of Thesis

Overview of Chapter 1: This Chapter starts with an introductory section covering a synopsis of the thesis. This is followed by a literature review of the genetic algorithm (GA) technique as an optimisation tool in various digital signal processing (DSP) applications. The generic form of the GA code used for optimisation in this study is then described. Finally, the aims, objectives and contributions of this study are considered.

1.1 Introduction

This thesis is concerned with the issues of design and optimisation of digital filters and multirate filter banks. The main focus and contribution of this thesis is to apply the genetic algorithm (GA) technique and to draw some comparison with the standard gradient and non-gradient based optimisation methods. The accuracy of a real-time digital filter frequency response is affected by the finite word length (FWL) constraint used in its implementation. The full process of digital filtering can generate errors in a number of ways such as; quantisation of the input signal due to analogue to digital conversion, representation of filter coefficients by a finite number of bits and the accumulation of round-off errors resulting from arithmetic operations. Overflows of arithmetic operations can also occur within the filtering algorithm, however, with proper scaling procedures, such overflow problems are easily eliminated.

For the case of digital filters, this study is concerned with the optimisation of finite word length coefficients using genetic algorithms. Both non-recursive and recursive filters are considered. It is well recognised that structural differences of digital filter implementation can have significant variations in their sensitivity to filter response as a consequence of coefficient truncation due to finite word length effects.

The linear phase, direct form structure of finite impulse response (FIR) filters has been shown to be robust and therefore, attractive for the realisation of FWL coefficient implementation [Chan and Rabiner, 1973]. The problem of FWL FIR symmetric digital filters involves choosing a set of coefficients so that the new frequency response, as a consequence of truncation of the infinite precision coefficients, approximates as closely as possible to a given specified frequency response in a minimax sense. Algorithms for solving this problem have been based upon two methods; the local search method [Avenhaus, 1972] and the integer programming 'branch and bound' method [Kodek, 1980], [Kodek and Steiglitz, 1981].

The local search algorithm involves selecting a feasible set of FWL coefficients (say rounded valued) to give a frequency response and examining the neighbourhood of H, the transfer function of the filter, for a better filter H' i.e. one with lower error function. If such a filter is found then H', replaces H and the algorithm moves to the next step or else it stops. The 'branch and bound' algorithm is involved with systematically pruning a tree of several possible solutions based upon certain lower bounds as the enumeration proceeds. Both of these methods are intrinsically computationally intensive and global optimality is not assured. The problem is further compounded and becomes acute for longer filter lengths.

For the case of infinite impulse response (IIR) filters, the direct form implementation is usually avoided. It has been shown [Kaiser, 1966] that the sensitivity of the filter response to the FWL effects of the denominator coefficients in this case increases rapidly with the increasing order of the filter. A 2^{nd} order cascade or parallel form implementation of IIR filters is better behaved in this situation [Dempster and Macleod, 1994].

For the case of multirate filters, a 2-channel quadrature mirror filter bank (QMF) and multiple-channel uniform and non-uniform filter banks have been considered in this study. Multirate systems are based on the application of digital filters for which sampling rate can vary from point to point. Such systems are often used for processing signals more efficiently and have seen applications in the sub-band coding of speech, audio and video signals and in multi-carrier data transmission and digital audio systems. A number of different design and implementation structures exist for multirate filter banks [Vaidyanathan, 1993]. Emphasis is placed, in this study, on the choice of structure that is relatively simple to design and leads to an efficient practical implementation for real-time applications.

All multirate filter banks suffer from the problems of amplitude, phase and aliasing errors and therefore, constraints for perfect reconstruction (PR) of the input signal can be extensive. In addition, coding errors are generated by the quantisation of the decimated signal that cannot be eliminated. These coding errors are beyond the scope of this work and will not be considered in this study. However, the first three errors can be reduced and almost eliminated by prudent choice of design criteria and type and order of sub-band filters leading towards perfect reconstruction of the input signal.

The case of a maximally decimated, two-channel filter bank is the simplest example of a multirate filter bank. This is commonly referred to as a quadrature mirror filter (QMF) bank since the high pass filter is a mirror image of the low pass filter about the mid-point. The 2-channel QMF bank considered here for optimisation is designed using the *transformation of variables* method [Tay, 1998]. This design technique yields IIR filters for sub-bands that are causal and stable and can achieve perfect reconstruction of the input signal. The design of such a filter bank is based on the use of prototype filters whose variables are transformed using a transformation function. This design is fairly simple and allows flexibility for 'fine-tuning' of

the design and implementation for real-time realisation. However, a specific problem that exists in this method of design is to do with optimising the parameters of the transformation function to generate a good parity between the idealised and the actual frequency response of the filters. This process involves the use of optimisation procedures such as the quasi-Newton or downhill Simplex algorithms. Most standard procedures of this form are inevitably dependent on the selection choice of the starting 'seed' values of the transformation function parameters, so a good optimal result cannot always be assured in such situations.

A genetic algorithm based optimisation approach has been considered in this work to search for the global minima over a wide landscape of transformation function parameter values. A new GA 'creep' code has also been developed here to study the effects of small variations of the parameter values once a 'suspect' good valley has been detected by the main GA code. This 'creep' code is a variation of the G-bit operator [Goldberg, 1989]. A comparative study of the GA based hybrid optimisation (GA optimisation followed by standard quasi-Newton and downhill Simplex methods) and non-GA based standard optimisation methods is also conducted. In addition, FWL constraints are applied to the optimisation of IIR filters using a genetic algorithm code. This leads to the development of practical IIR sub-band filters. The optimised QMF filter structure is represented in a computationally efficient form using polyphase decomposition and tested using simulation and a real-time DSP system.

The second form of multirate filter considered in this work is a maximally decimated multiple M-channel uniform filter bank. This form of filter bank consists of equally spaced frequency bands that are each of equal widths. Such a structure offers a higher resolution for the analysis of the input signal and can be an improvement over the 2-channel QMF bank. Closed form solutions for the design of the M-channel filter bank without aliasing and with perfect reconstruction property are well established [Vaidyanathan, 1993]. Mostly such solutions lead

to complex implementations. In practice, therefore, approximations that consider only the directly adjacent alias spectra and its minimisation are less complex and thus of greater interest. These types of filter banks are called pseudo-QMF banks and use cosine modulation technique for its design, based on a single prototype low-pass filter. The optimisation of the prototype filter such that the signal reconstruction errors are minimised forms an important area for consideration that is investigated in this study.

The third form of multirate filter studied here is the non-uniform filter (NUF) bank. This type of filter bank has specific advantages in their application to real signals for which high coding gain is achievable. This is specifically due to the significant variation of the ensemble average of the energy in different frequency bands of real signals. The case for multiple-band uniform filter banks, as mentioned above, has been extensively studied and PR conditions are well established [Vaidyanathan, 1993]. However, the design of non-uniform banks is particularly challenging for PR for which extensive constraints exist. The problem, in general, is reduced to relaxing constraints at the expense of errors and finding methods for minimising the errors. Optimisation techniques are thus commonly used for the design and implementation of non-uniform filter banks and are investigated in this study.

Several examples of design and optimisation of NUF banks have been reported in literature recently. The concept of compatible sets for integer decimation factors as a requirement for completely eliminating aliasing error in a maximally decimated NUF bank is reported in [Hoang and Vaidyanathan, 1989]. Other examples of design are based on the use of optimised multiple-prototype LP filters and cosine modulation or by sine/cosine multiplication to shift to the appropriate frequency sub-band. The optimisation process is aimed at eliminating the main aliasing component [Argenti and Del Re, 1996], [Wada, 1995]. Another approach is based on time-domain analysis for the design of NUF banks using FIR filters [Nayebi *et al*, 1993] .

A direct design approach using FIR low pass prototype (baseband) filters and their transformation using sine or cosine multiplication is considered in this study [Chu, 1985], [Wada,1995]. The case of a non-compatible, integer-valued, maximally decimated set NUF bank is investigated. This form of filter bank gives greater flexibility for a choice of sub-band filters for the optimisation of coding gain of a real signal at the expense of aliasing and amplitude distortion that cannot be completely eliminated. Effort is then applied to reducing the two distortions by using a hybrid approach based on a genetic algorithm and standard minimisation techniques such as quasi-Newton and downhill Simplex.

1.2 Genetic Algorithms in DSP design and optimisation

Genetic algorithms are intensively parallel stochastic search algorithms based on the principles of natural genetics and the concept of 'survival of the fittest'. GAs operate over a wide landscape of search space using a large population set of possible solutions seeking to locate potentially the best candidates. The process of ranking, crossover and mutation is applied sequentially through a number of generations in an effort to obtain a good optimal solution. The characteristics of GAs makes it a versatile tool to conduct a search over a large, noisy, multi-modal and possibly a discontinuous, search space landscape. For continuous and slowly varying landscapes, then the standard calculus based methods are likely to do better, however, for smaller landscapes, the GAs may show no specific advantage over the enumerative or random search methods [Holland, 1975], [Goldberg, 1989].

The GA used in this study is a MATLAB based programme that was developed for the study of control systems [Chipperfield *et al*, 1993]. Although this version is fairly flexible in its implementation to different applications, only real-valued simple GAs are used for the work

covered in this study. The use of real-valued genes has specific advantages in the numerical objective function optimisation over the binary-coded GAs [Wright, 1991]. The GA efficiency is increased since there is no need for conversion of chromosomes to phenotypes before the objective function is evaluated. The other advantages are; less memory overhead for computation, no loss of precision by discretisation to binary and greater freedom for the use of different genetic operators.

Design of digital processing systems based on minimal computational complexity and low power has important implications in modern systems especially for portable applications. Inevitably, for many instances, there is a need for optimising the hardware structure and realisation of processes involved. The use of primitive filter components such as simple adder sections represented by their corresponding transfer functions of the form $H(z) = 1+z^{-k}$, where k is an integer, can be stored as library entries. A combination of these primitive filters represents a trial frequency response that is compared to the specified response and then the error function calculated. The optimisation of a combination of filter primitives can lead to the design of FIR filters of minimal computational complexity. A GA based optimisation using a trial selection of a combination of filter primitives can lead to an efficient method of FIR filter design generating filters of good frequency response and minimal computational complexity [Suckley, 1991]. Suckley has used a sequential GA to design a cascaded FIR filter structure that optimises the objective function metric based on computational efficiency of the filter. Primitive filter components, as defined by Wade [Wade *et al*, 1990], are used as *genes* and cascaded primitive structures represented a *chromosome*. A direct design method for FIR digital filters with arbitrary log magnitude and phase responses based on obtaining a least-squares approximation using a weighted genetic algorithm for each iteration is reported in [Lu and Tzeng, 2000].

Another efficient form of FIR filter realisation exploits the redundancy that exists in the product terms of the convolution form representation of such filters in the time domain. The partial results of a single multiplication process of a coefficient and a signal sample value can be reused to assist in the formation of other product terms. Thus, each inner product in the convolution process can be decomposed into a number of primitive arithmetic operations such as additions, subtractions and shifts. This can lead to an efficient architecture that comprises an interconnection of elementary digital processing elements. The representation of an FIR filter multiplication block in the form of an efficient signal flow directed graph using primitive operators is a useful methodology for direct form FIR digital filter realisation both in terms of its frequency characteristic and hardware implementation [Bull and Horrocks, 1991]. Genetic algorithms have been used for the efficient implementation of primitive operator directed graphs to provide FIR filter design in consideration of compromises between filter performance, complexity and filter order [Redmill and Bull, 1997, 1998]. The use of a hybrid GA optimisation for the direct design of frequency selective FIR digital filters based on the frequency sampling method is reported in [Harris and Ifeachor, 1998].

The problem of approximating the digital filters designed using infinite precision coefficients by using a finite number of bits for real-time realisation has been mentioned in section 1.1. The frequency response characteristic of such filters deviates from the original by a finite amount. Another issue here is the representation of coefficient values by a significantly reduced number of bits so that the filter implementation can be achieved with much improved computational throughput. This effort can lead to reduced complexity architecture with savings in memory, space and power [Arslan and Horrocks, 1995]. Another work investigated in this area for designing FIR digital filters is based on the minimisation of the minimax criterion and the normalised peak ripple magnitude using genetic algorithms [Ciloglu, 2002].

The results of integer programming method for optimising finite word length coefficient FIR filter frequency response have been compared with those obtained using a parallel GA [Xu and Daley, 1995]. The parallel GA is executed using 16 microprocessors and implemented using parallel 'C' language. This effectively involves running 16 sequential GAs running simultaneously and linked by the operation of 'migration'. The optimisation metric based on weighted pass-band and stop-band deviations showed marginal improvements using the GA procedure. Another application of GAs is reported for the case of direct design approach for IIR digital filters that consider a hierarchical multi-layer gene structure to represent a chromosome. This chromosome structure is formed using a string of control genes representing the filter structure that is concatenated with another string of genes representing the filter coefficients. The design process attempts to satisfy constraints of; lowest order filter, speed of computation, filter stability and frequency response tolerance settings [Tang *et al*, 1998].

The synthesis of VLSI low power hardware design specifically for digital signal processing applications is another area of interest for modern portable communications and computing systems. Issues of space, speed and power are important factors in such systems. A high-level signal data flow graph (DFG) consisting of functional blocks such as adders, multipliers and delays can be used to formulate the DSP design [Bright and Arslan, 2001]. The data flow graph is encoded as an individual gene in the chromosome that may be of a variable length. The fitness function used here is based on minimising the overall power consumption that is dependent on the power consumption of individual DFG gene. In the area of multirate quadrature mirror filter banks, a recent work reported in literature uses a genetic algorithm for the optimisation of a canonical signed power-of-two (SPT) coefficient lattice structure. The genetic operations are constrained such that the canonical property of the SPT is preserved [Yu and Lim, 2002].

1.3 Genetic algorithm used in this work

The GA used in this work was developed for control systems based on the Matlab programme [Chipperfield *et al*, 1993]. The simple GA generic code used in the work described in the thesis, remains largely the same. Figure 1.1 shows the generic code. The 'ranking' function in the code returns a column vector based on the corresponding individual fitness values and then ranks the individuals for minimisation of the objective function. The option selected performs a linear ranking with a selective pressure (SP) of 2 [Whitley, 1998]. The fitness value assigned to the individuals is calculated according the following formula:

$$FitnV(Pos) = 2 - SP + 2(SP-1)(Pos-1)/(Nind-1) \qquad 1.1$$

Where 'Pos' is the position of the individual in the sorted population and 'Nind' is the number of individuals used.

The high-level function for selection of individuals from the population set and returning the selected individuals in a new population is performed by the 'select' function. The low-level selection function 'sus' is called by the 'select' function. The 'sus' function is based on a form of stochastic sampling method and is implemented by obtaining a cumulative sum of the fitness vector, FitnV, and generating a set of equally spaced numbers between 0 and Σ(FitnV) [Baker, 1987]. The probability of an individual being selected is given by:

$$F(x_i) = \frac{f(x_i)}{\sum_{i=1}^{Nind} f(x_i)} \qquad 1.2$$

where $f(x_i)$ is the fitness of individual x_i.

```
NIND = number of individuals; MAXGEN = maximum number of generations;
GGAP = generation gap; INSR = Insertion rate; N = number of sample points;
% Build a field descriptor
FieldDR=define search space
% Create a real-valued initial population
Chrom=crtrp(NIND,FieldDR);
% Evaluate initial population
ObjV=obj_function(Chrom,lb,lc,M,N,n);
gen=0;              % Counter
% Generational loop
while gen < MAXGEN
        % Assign fitness values to entire population
        FitnV = ranking(ObjV);
        % Select individuals for breeding
        SelCh=select('sus', Chrom, FitnV, GGAP);
        % Recombine individuals (crossover)
        SelCh=recombin('recdis',SelCh);
        % Apply mutation
        SelCh=mutbga(SelCh,FieldDR);
        % Evaluate offspring, call objective function
        ObjVSel=obj_function(SelCh,lb,lc,M,N,n);
        % Reinsert offspring into population
        [Chrom ObjV]=reins(Chrom,SelCh,1,[1 INSR],ObjV,ObjVSel);
        % Increment counter
        gen=gen+1
end
```

Figure 1.1: A generic Simple GA code in MATLAB

The crossover function is also performed in two stages. The high-level function is 'recombin' that calls the low-level function 'recdis'. A number of low-level recombination (crossover) options are available. The single or multiple point crossover functions (with and without shuffel) are applicable mainly to binary-type variables. The 'recdis' function used in the work described in the thesis is a discrete recombination function. The mating process is performed between pairs of rows. The 'recdis' function first generates an internal mask table that determines which parents contribute which variables to the offspring. On the basis of the randomly generated mask table, the variable values are exchanged between the individuals. The mutation operator is represented by 'mutbga' that takes real-valued population, mutates each variable with a pre-defined probability and returns a new population after mutation. The 'mutbga' function produces firstly a random internal mask table that determines which variables will mutate and also the sign for the step size. A second internal table generates the normalised mutation step size. The mutated variable is worked out as a function of the original variable and the step size [Muhlenbein and Schlierkamp-Voosen, 1993]. Finally, the 'reins' function performs insertion of the offspring into the current population replacing the parents and returning a new population set.

1.4 Aims and Objectives

The main aim of this work is to assess the genetic algorithm technique as an optimisation tool in a selected range of digital signal processing (DSP) applications. More specifically, in the field of digital filters, the problem of finite word length (FWL) coefficients is considered for optimisation. In the field of multirate filter banks, the optimisation of the design issues and error problems for perfect reconstruction of the input signal, are considered using genetic algorithms.

The objectives of this work comprised the following.

1. To investigate the feasibility of using a simple sequential GA as an optimisation tool for a selection of digital signal processing applications and to draw a comparison with the results reported in literature where this is possible.

2. To use the hill climber optimisation techniques such as steepest ascent (SAHC), nearest ascent (NAHC), downhill Simplex, quasi-Newton and the sequential quadratic programming (SQP) constrained optimisation method for the purpose of drawing a comparison with the GA optimised results.

3. To test the validity of the GA technique as an optimisation technique in the design of multirate filter bank by using it in a hybrid form such as GA optimisation followed by a standard gradient and non-gradient based hill climber technique.

The motivation for using genetic algorithms to the design and realisation of digital filters and multirate filter banks was the need to generate near-optimal solutions by searching over a fairly wide landscape of possible solutions without using computationally intensive and slow, iterative techniques. It is believed that the work described in this thesis is the first time that a quantifiable measure of performance improvement using GAs for FWL coefficient digital filters has been made. This measure is based on a comparison with the results of the filter frequency response obtained using simply rounded coefficients and, for the case of FIR filters, also with certain statistical bounds arrived at by mathematical analysis of the quantised coefficients [Chan and Rabiner, 1973]. Further comparison of the GA optimised results for FWL coefficient digital filters is based on the optimised results using the 'steepest' and 'near' ascent hill climber techniques [Mitchell, 1996].

It is also believed that GAs have been used for the first time in the optimisation of multirate filter banks. For the case of a 2-channel uniform quadrature mirror filter bank, the contribution of the work in this thesis is the GA optimisation of the parameters of the transformation function required in the design of the QMF bank using the *transformation of variables* method [Tay, 1998]. This leads to the determination of optimal coefficients of the IIR analysis and synthesis digital filters. Furthermore, the FWL coefficients are optimised using a second stage GA. The optimised filter bank is first tested using the SIMULINK Matlab programme and then implemented on a TMS320C50 fixed point digital signal processing kit [TMS 320C5x DSK, 1997] for real-time testing for a number of coding options.

For the case of non-uniform filter banks, significant constraints exist for satisfying the perfect reconstruction property. However, for real signals, the ensemble average of energy varies significantly in different frequency bands that do not easily conform to the requirements of the constraints for PR. A number of constraints are thus relaxed causing significant amplitude and aliasing distortions. It is believed that this thesis contains the first explicit method using GAs, for optimising the overall non-uniform filter bank transfer function by perturbing the cut-off frequencies of the prototype low-pass filters individually. This procedure overcomes the problem of reconstruction errors by reducing the amplitude and aliasing distortion in a combined manner. Some design examples of a maximally decimated, non-compatible set, for non-uniform filter banks using FIR filters have been considered and tested using Simulink toolbox of Matlab.

1.5 Overview of the thesis and the contributions

The structure of the thesis follows the description of the work covered in section 1.1 above. An introduction to the genetic algorithm used for optimisation in the main body of the work is followed with the theoretical and optimisation issues for the various digital filtering processes. The FIR and IIR digital filters are first investigated followed by the uniform multirate quadrature mirror filter bank. An investigation of the modal property of the QMF bank objective function is conducted and a comparative analysis of the performance of different optimisation procedures is also considered. This is followed with the investigation of a class of uniform and non-uniform multirate filter banks and their optimisation using GAs. Finally, the conclusions from the work covered and described in the thesis are drawn and recommendations made.

Chapter Two describes the finite word-length coefficient problem of FIR filters and discusses issues of statistical bounds. The objective functions are defined for various filter types such as low-pass, high-pass etc. and GA optimised results are compared with the results obtained using the integer programming method. Chapter Three discusses aspects of IIR filter design and structure type such that coefficient finite word-length effects are minimised. Stability issues are also discussed and considered in the GA optimisation of such filters. Since coefficient finite word length constrained statistical bounds are not known for IIR filters, a comparison of GA optimised results is made with the simply rounded-valued coefficients of the filter. A range of FWL coefficient bits starting from a high number of bits to a fairly low number, have been used to test the advantage of the GA optimised results compared to the simply rounded coefficients.

Chapter Four reviews the theoretical issues of the *transformation of variables* design of the quadrature mirror filter bank with perfect reconstruction property. The analysis and synthesis

filters are designed using causal, stable IIR filters that exhibit a linear phase characteristic. The GA optimisation of the parameters used for the transformation is also covered here including a comparative analysis of the optimisation results with and without the use of GAs. Further study in this chapter covers the aspects of practical implementation of the optimised QMF bank. The computationally efficient method of polyphase decomposition of the QMF bank is discussed and also used in the implementation both in the simulated results and for real-time testing on the TMS302C5x digital signal processing hardware kit. A number of possible coding gain options are implemented and tests conducted using the 'mean opinion score' procedure.

In Chapters Five and Six, the theoretical issues of multiple M-channel uniform and non-uniform, multi-band multirate filter banks are discussed respectively. The constraints for perfect reconstruction are also investigated and some of these constraints are relaxed. This leads to the generation of amplitude and aliasing distortions that are minimised by using a hybrid approach combining firstly the a GA search over a wide landscape and then using the standard quasi-Newton and downhill Simplex method on the most promising near-optimal solution. The optimised examples of the uniform and non-uniform filter banks based on FIR filters are tested using the Matlab Simulink package. Finally Chapter Seven reviews the thesis and summarises the conclusions from the work presented in the thesis. It also makes recommendations for further work. Appendix A contains copies of four conference papers that have been published during the course of the work described in this thesis.

The main focus and contribution of this thesis is the study and application of the genetic algorithm optimisation method in the area of digital filters and multirate filter banks. More specifically, the following contributions are claimed.

- *A real integer-valued genetic algorithm code has been developed for the optimisation study of finite word length constrained coefficients of FIR digital filters. Some comparative study has been investigated and reported.*

- *Real integer-valued genetic algorithm codes have been developed for the optimisation of the finite word length constrained coefficients of IIR digital filters. The direct form and the second order cascade form structures have been considered and extensive range of new results obtained.*

- *A real-valued genetic algorithm code has been developed for the optimisation of the design of a class of quadrature mirror filter bank that has a perfect reconstruction property. For a comparative study with other standard methods, this GA code was further enhanced to include a 'creep' code option within the main GA code that uses a 'tumbling-like' minimisation algorithm.*

- *The new GA optimised design of the QMF bank was implemented on a real-time TMS320C50 digital signal processing starter kit. Tests were conducted using the Mean Opinion Score metric for telephone quality signals.*

- *A real-valued genetic algorithm code has been developed for the optimisation of design of a uniform maximally decimated M-channel filter bank. The process involved marginally perturbing the prototype filter parameters for optimal results.*

- *Real-valued genetic algorithm codes have been developed for the optimisation of the non-uniform M-channel maximally decimated filter banks using integer decimators. Multiple low pass prototype filters are used in the design stage. The hybrid optimisation process is*

applied to the entire network of the non-uniform filter bank. The minimisation of the magnitude and aliasing errors is thus achieved in a combined manner.

Chapter 2: Finite word length optimisation of FIR filters

Overview of Chapter 2: This Chapter starts by considering the filter coefficient finite word length (FWL) problem in regard to the finite impulse response digital filters. Some theoretical issues and statistical error bound conditions of the maximum deviation between the exact and the approximate magnitude responses are also considered. The GA optimisation results for the maximum error bounds and error deviation due to FWL effects for a number of design examples are investigated. Finally, a comparison is drawn between the simply rounded, the GA optimised, integer programming method and the simple hill climber methods.

2.1 Introduction

The work described in this chapter starts with the discussion on the specific issues of finite word length constraint for digital filters in general and more specifically for the case of finite impulse response (FIR) filters. The analysis of simple and statistical bounds for FWL coefficients of FIR direct form filters is reviewed and used for a comparative study with the GA optimised results. A number of objective functions for the various structures of the FIR filter such as low-pass, high-pass etc. are also defined with the view to practical significance of the optimised frequency response. The optimisation problem of FWL coefficients of FIR filters obtained by rounding the 'infinite precision' coefficients has been investigated previously [Avenhaus, 1972], [Kodek, 1980], [Kodek and Steiglitz, 1981]. A comparative study for the case of a FIR filter coefficient optimisation using the integer programming method and GA optimised results using the specific code developed for the work in the thesis is covered in this chapter. Furthermore, a small selection of filters was tested using the simple hill climber techniques and the optimised results are compared with the GA optimised results.

2.2 The finite word length problem

The finite word length effect in digital filter implementation causes errors in their realisation in a number of ways. The analogue to digital conversion of the input signal causes quantisation noise that is well understood as considered in pulse code modulated systems. Using additional number of bits to represent sampled signal values can reduce the effect of these errors. The filter coefficient quantisation errors are due to representing the filter coefficients with a limited number of bits for real-time, fixed point processing of the signal. This error has an adverse effect on the transfer function of the filter and in some instances the frequency response may vary significantly from its original form (see Figure 2.1). The third form of error is due to the round-off errors in the arithmetic operations. This occurs by discarding say the lower order bits before storing the results of multiplication. The effect of this error is the reduction of signal to noise (SNR) ratio and is constrained by the type of arithmetic used. Lastly, the arithmetic overflow problem occurs when partial sums or filter output exceeds the maximum number of bits allowed by the system. This results in an incorrect interpretation of the output sample value. This form of error is overcome by appropriate scaling of the coefficient values.

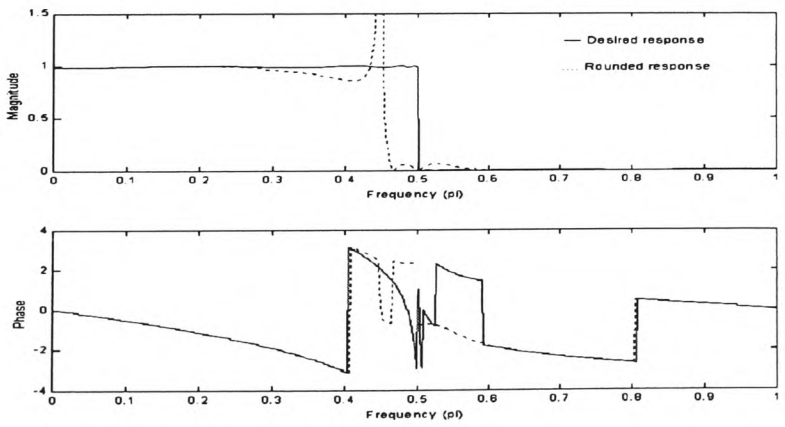

Figure 2.1 Effect of coefficient quantisation on frequency response of digital filters.

The error due to coefficient quantisation can be reduced in practice by applying optimisation techniques that allow for efficient means by which the quantised coefficients are perturbed by a small amount that leads to a closer approximation of the new frequency response to the original response. The work covered in this thesis considers this type of problem and a study of the use of genetic algorithm as an optimisation tool is investigated by using a number of metrics for a comparative analysis.

2.2.1 Finite word length coefficient effects in FIR filter realisation

The first stage in the realisation problem of a FIR digital filter is the design of the filter that approximates the original specifications. This process leads to the calculation of high precision filter coefficients in the transfer function of the approximated filter. The second stage of the filter design involves its realisation either on a digital hardware system or as a software programme to implement the input/output relationship as prescribed by the filter transfer function. For a given transfer function, there may exist many different forms of structures for implementing and/or programming the digital filter. In the case of FIR filters there are a number of different types of filter structures of which the most commonly used is the direct form. Others are; fast convolution, frequency sampling, transpose and cascade structures. The choice for selecting a particular structure for a specific application can depend on several factors. These are; sensitivity to errors in the filter coefficients, ease of programmability for a particular processor, immunity to signal quantisation etc.

A simple 3-length FIT filter can be represented by a transfer function H(z) given by

$$H(z) = h_0 + h_1 z^{-1} + h_2 z^{-2} \qquad 2.1$$

Where h_0, h_1, h_2 are the filter coefficients and z^{-1} represents one unit time delay.

The difference equation that provides the output condition is given by the convolution operation of

$$y(n) = h_0 x(n) + h_1 x(n-1) + h_2 x(n-2) \qquad 2.2$$

The direct calculation of Equation 2.2 can be represented by a block diagram shown in Figure 2.2 and is called the direct or transversal structure. This form of structure is most commonly used in practice due to its robustness and its ease of programming and efficient implementation on most DSP devices. Such DSP devices are specifically designed and include machine code instructions that are tailored for efficient FIR transversal operations [Ifeachor and Jervis, 1993], [Parks and Burrus, 1987]. For this reason, only the direct form structure is considered for study in this work.

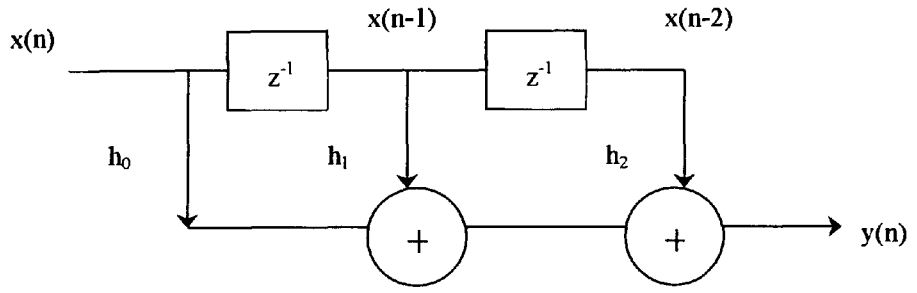

Figure 2.2 Direct form FIR filter structure of length 3.

The calculation of the output condition of the FIR filter given by Equation 2.2 involves working out the products of the coefficient values and the past and present input signal values. In practice, real-time implementation and realisation of FIR filters is often involved with the use of fixed-point digital devices that are designed for optimal throughput of FIR filtering operations. This condition imposes a restriction on the number of bits that can be used to represent the signal data value, the filter coefficients and the results of arithmetic operations. For efficient computational throughput and to limit the cost of the DSP device, the number of bits used to

represent the various values must be small. This restriction leads to the problem of finite word length effects in the filter realisation and in general, degrades the performance of the filter when compared with the original design. The study in this thesis considers the specific problem of FWL coefficients and seeks to optimise the frequency response of the filter by picking the best-quantised coefficients. The realisation of FWL FIR filters is normally executed on fixed point DSP devices that are usually cheaper than their floating-point counterparts. There are also advantages for fixed-point devices in terms of smaller silicon space, less number of external pins, lower power dissipation and faster clock cycle times. Such fixed point FWL devices find applications in digital audio systems, speech processing and compression and in mobile communications. The study in this chapter is thus restricted to the use of quantised coefficients for fixed-point devices and the calculations involved are based on fixed-point arithmetic.

The most commonly used method of deriving FWL coefficients for fixed-point arithmetic is the direct quantisation method. In this method, the high precision coefficients that are derived using standard filter design techniques are first rounded to yield FWL quantised coefficients. The starting solution of quantised coefficients is thus given by

$$h_{ri} = \text{round}[h_{ei}\, 2^{B-1}] \qquad i=0,1,2,\ldots\ldots,N-1 \qquad 2.3$$

Where 'h_{ri}' is the rounded coefficient, 'h_{ei}' is the high precision coefficient, 'B' is the number of bits used to represent the coefficients and 'N' is the filter length. The representation of a high precision coefficient value of say 0.762345 in a 5-bit rounded form is thus given by h_r = round[0.762345 x 2^4] = round[12.19752] = 12. This value, in a fraction form is represented by $12/2^4$ = 0.75. Thus a new set of coefficient values are derived that generate the 'rounded' frequency response of the filter. It must also be noted that for fixed-point devices, two's complement arithmetic is most commonly used and the most significant bit in the above

example of a 5-bit representation is the sign bit. The largest positive number is then +15 and the largest negative number is −16. This form of numbering system is the only one used and considered in this work mainly due to its application later (Chapter 4) on a fixed-point DSP device based around Texas Instrument's TMS320C50 processor that stores numbers in a two's complement integer format. Furthermore, the work of Kodek and Steiglitz [1981] that has been used for comparative purposes in this chapter also use the two's complement integer format for representing the coefficient values of the designed FIR filters.

2.2.2 Finite word length coefficient quantisation

The exact representation of a direct form non-recursive or FIR filter is of the form:

$$y(n) = \sum_{m=0}^{N-1} h(m)\, x(n - m) \qquad 2.4$$

where $x(n)$ = input signal

$y(n)$ = output signal

$h(m)$ = coefficient value of the filter

and N = filter length (total number of coefficients)

For quantised values of the coefficients then the output signal is modified to:

$$\hat{y}(n) = \sum_{m=0}^{N-1} \hat{h}(m)\; x(n - m) \qquad 2.5$$

where $\hat{h}(m)$ = quantised coefficient value

The error signal e(n) is then given by

$$e(n) = y(n) - \hat{y}(n) \qquad 2.6$$

or

$$e(n) = \sum_{m=0}^{N-1} \{h(m) - \hat{h}(m)\}\; x(n - m) \qquad 2.7$$

In terms of the frequency response, the system transfer function with quantised coefficients is given by

$$\hat{H}(\omega) = \sum_{m=0}^{N-1} h(m)\, e^{-j2\pi f_m} + \sum_{m=0}^{N-1} \{h(m) - \hat{h}(m)\}\, e^{-j2\pi f_m} \qquad 2.8$$

or $$\hat{H}(\omega) = H(\omega) + E(\omega) \qquad 2.9$$

where $E(\omega)$ is the transfer function of the error signal and $H(\omega)$ is the transfer function of the unquantised exact system.

The magnitude of the frequency response of the error signal is bounded by the inequality

$$|E(\omega)| \leq N \max |\{h(m) - \hat{h}(m)\}| \qquad 2.10$$

When the coefficients are rounded to B bits, including the sign bit, then

$$\max |\{h(m) - \hat{h}(m)\}| = 2^{-B} \qquad 2.11$$

then, $$|E(\omega)| \leq N\, 2^{-B} \qquad 2.12$$

Also note that the bounded value of Equation 2.12 has been derived for the case of arbitrary phase FIR filter i.e. no coefficient symmetry is assumed. However, a symmetrical spread of the coefficients taken from the central point is a typical characteristic of linear phase FIR filters. This is a special case for the derivation of error bound of Equation 2.12 and therefore, the same error bound as given by Equation 2.12 also applies to the case of linear phase FIR filters.

The upper bound of the magnitude of error signal from Equation 2.12 gives a worst case limit and is overly pessimistic. For this reason, statistical error bounds have been developed that give a more realistic measure of the bounded limits [Chan and Rabiner, 1973]. The statistical analysis of the filter response errors is based on the assumption that errors due to different

coefficient quantisations are statistically independent and that each error is uniformly distributed in the range –Q/2 and +Q/2 where Q is the quantisation step size. This assumption leads to the deduction that the error signal will have a zero mean and a variance of $Q^2/12$ which is consistent with rounding of sampled signal values in pulse code modulated systems. Two error bounds are derived based on statistical analysis. These are:

i) for linear phase FIR filter

$$\sigma_{EL}(\omega) \leq \frac{Q}{2}\sqrt{\frac{2N-1}{3}} \qquad 2.13$$

ii) for arbitrary phase FIR filter

$$\sigma_{EA}(\omega) \leq \frac{Q}{2}\sqrt{\frac{N}{3}} \qquad 2.14$$

where $\sigma_{EL}(\omega)$ and $\sigma_{EA}(\omega)$ are the standard deviation of the errors respectively.

2.2.3 Some practical considerations for filter design

Filter design problems in general, involve finding a filter with a frequency response, which approximates that of an ideal filter response within a specified amount of error. More specifically, for band select filters, several bands may be defined for which error bounds are also specified. In this part of the study, only band select filters will be considered although, this restriction does not in any way affect the possibility of generalising the optimisation ideas and test metrics used for a comparative study.

Let $D(\omega)$ be some real, idealised band-select function that is desired to be approximated by the exact frequency response of a linear phase FIR filter denoted by the transfer function $H(\omega)$. The desired function $D(\omega)$ consists of a number of disjointed frequency bands $\Omega_k \subset [0,\pi]$, where k=1,...,M such that for each k, $D(\omega)$ is to be approximated to within a specified error

bound $\delta(k)$ for all $\omega \in \Omega_k$ The frequency band Ω_k is separated by M-1 transition bands where the filter frequency response remains unconstrained. This assumption can lead to severe excursions from the high-precision design of frequency response $H(\omega)$ when coefficient quantisation, especially low-bit form of representation is used. This effect has been demonstrated in an example of the optimised response shown later in this section.

For $D(\omega)$ to be the desired response in the specified frequency bands, then

$$\text{Max}\left| \, |H(\omega)| - D(\omega) \, \right| = \delta(k) \qquad \text{for } \omega \in \Omega_k \tag{2.15}$$

Now for all ω

$$\left| \, |\hat{H}(\omega)| - D(\omega) \, \right| \leq \left| \, |H(\omega)| - |\hat{H}(\omega)| \, \right| + \left| \, |H(\omega)| - D(\omega) \right| \tag{2.16}$$

thus

$$\left| \, |\hat{H}(\omega)| - D(\omega) \, \right| \leq \max_{\omega \in \Omega_k} |E_L(\omega)| + \delta(k) \tag{2.17}$$

where $E_L(\omega) = |H(\omega)| - |\hat{H}(\omega)|$

Also, it is assumed that in all probability

$$|E_L(\omega)| \leq 2\sigma_e \tag{2.18}$$

where $\sigma_e = \max \sigma_{EL}(\omega) = \dfrac{Q}{2}\sqrt{\dfrac{2N-I}{3}}$

then,

$$\left| \, |\hat{H}(\omega)| - D(\omega) \, \right| \leq Q\sqrt{\frac{2N-1}{3}} + \delta(k) \qquad \text{for } \omega \in \Omega_k \tag{2.19}$$

or

$$\max \left| \, |\hat{H}(\omega)| - D(\omega) \, \right| = Q\sqrt{\frac{2N-1}{3}} + \delta(k) \qquad \text{for } \omega \in \Omega_k \tag{2.20}$$

Where $Q = 2^{-B}$ and B is the number of bits including the sign bit. Note that the upper bound of Equation 2.20 is valid provided that the infinite precision filter conforms to the design

requirement stipulated in Equation 2.15. For conducting a comparative study, the metric of Equation 2.20 over the selected frequency bands will be considered. A comparison is drawn between the GA optimised and the integer programming method of optimisation.

2.3 Objective functions for filter optimisation

The generalised objective error function 'F' to be minimised can be represented in the form

$$F = \sum_{i=0}^{L} (| H_{ei} - H_i |^2 \; w_i) + W \sum_{j=k}^{m} (| \Phi_{ej} - \Phi_j |^2) \qquad 2.21$$

where the first part of Equation 2.21 is the magnitude squared error and the second part is the phase squared error with W being a weighting fraction. It is also possible to apply weightings w_i to the individual magnitude errors at frequencies $i\pi/L$ for i = 0, 1, 2, ..., L. Such individual weightings can be adjusted to give a better control for biasing the optimisation procedure towards the preferred design specification.

Also,

H_{ei} = Magnitude response of exact filter at frequency $i\pi/L$.

H_i = Magnitude response of finite word length filter at frequency $i\pi/L$.

Φ_{ej} = Phase response of exact filter at frequency $j\pi/L$.

Φ_j = Phase response of finite word length filter at frequency $j\pi/L$.

A different suitable objective function for the finite word length problem is

$$F = \sum_{m=0}^{L} | M(e^{j\omega_n}) - M_d(e^{j\omega_n}) |^2 \; W_m \qquad 2.22$$

where

$M(e^{j\omega_n})$ = complex response finite word length filter at frequency ω_n.

$M_d(e^{j\omega_n})$ = complex response of exact filter at frequency ω_n.

Here the objective function is the sum of the squared error of the complex response $M(e^{j\omega_n})$ compared to the desired response $M_d(e^{j\omega_n})$, over the frequencies $\omega_1, \omega_2, \ldots, \omega_L$ where $\omega_1 = 0$ and $\omega_L = \pi$. Note that the complex responses $M(e^{j\omega_n})$ and $M_d(e^{j\omega_n})$ contain both magnitude and phase information. Also, weightings W_m can be added for a proper bias of the optimisation procedure. Objective function of Equation 2.22 is biased towards finding an optimum solution both in magnitude and phase for the range between 0 and π and can yield a good solution in most cases. The objective functions of Table 2.1 offer more flexibility by allowing the removal of the stop-band frequencies for the phase response and hence relaxing the optimisation requirements. These can be used in difficult cases where Equation 2.21 fails to give an acceptable solution.

For FIR filters, phase linearity is ensured if the coefficients are symmetric around the mid-point coefficient(s). If the optimisation procedure of a FIR filter is designed to preserve the coefficient symmetry then there is no need to optimise for phase, because the phase response linearity will not be affected. Therefore, for the FIR case, the simple objective function shown in Equation 2.23 is adequate.

$$F = \sum_{i=0}^{L} | H_{ei} - H_i |^2 \, w_i \qquad 2.23$$

2.4 Optimisation of FIR filters using genetic algorithms

The Matlab programme `fir_ga.m` shown in Appendix B performs the GA optimisation of the direct form FIR filter by preserving the symmetrical form of the coefficients thereby generating linear phase response of the optimised filters. A utility programme `fir_obj.m` calculates the error objective function that is minimised over successive generations. Furthermore, the procedure for crossover, inherent in the discrete recombination operator 'recdis', was found to

Table 2.1

Objective function specifications

Filter type	Objective function	Parameters
Low-pass	$F = \sum_{i=0}^{L} (\lvert H_{ei} - H_i \rvert^2 w_i) + W \sum_{j=0}^{m} (\lvert \Phi_{ej} - \Phi_j \rvert^2)$	m: integer corresponding to the cut-off frequency ω_c $(0<\omega_c<\pi)$ $m \approx L\omega_c/\pi$
High-pass	$F = \sum_{i=0}^{L} (\lvert H_{ei} - H_i \rvert^2 w_i) + W \sum_{j=k}^{L} (\lvert \Phi_{ej} - \Phi_j \rvert^2)$	k: integer corresponding to the cut-off frequency ω_c $(0<\omega_c<\pi)$ $k \approx L\omega_c/\pi$
Band-pass	$F = \sum_{i=0}^{L} (\lvert H_{ei} - H_i \rvert^2 w_i) + W \sum_{j=k}^{m} (\lvert \Phi_{ej} - \Phi_j \rvert^2)$	k, m: integers corresponding to the lower the higher cut-off frequencies ω_{c1}, ω_{c2} $k \approx L\omega_{c1}/\pi$ $m \approx L\omega_{c2}/\pi$
Band-stop	$F = \sum_{i=0}^{L} (\lvert H_{ei} - H_i \rvert^2 w_i) + W \left(\sum_{j=0}^{m1} (\lvert \Phi_{ej} - \Phi_j \rvert^2) + \sum_{n=m2}^{L} (\lvert \Phi_{en} - \Phi_n \rvert^2) \right)$	m1, m2: integers corresponding to the lower the higher cut-off frequencies ω_{c1}, ω_{c2} $m1 \approx L\omega_{c1}/\pi$ $m2 \approx L\omega_{c2}/\pi$

be adequate for generating sufficient randomness in the population set so that the 'mutation' function in the programme was not of any benefit and thus not used in the main GA code for this application. The GA programme shown in Appendix B uses certain parameters that can influence the performance of the genetic algorithm and are explained below.

Peak variance 'BASE': BASE is an integer number indicating the maximum peak variance of the population individuals, compared to the original scaled and rounded coefficients. A value of 1 is suitable for most cases, but values of 2 or 3 or more can be used in some situations.

'Preserve Pattern' option: This option is activated when PRSZ = 1 and de-activated when PRSZ = 0. It is useful when optimising filters for which the zero-valued coefficients need to be preserved during optimisation.

Generation gap 'GGAP': If fewer individuals than the original population are produced by reproduction and crossover, then the fractional difference between the old and new population sizes is called generation gap. This can have a value between 0 and 1.

Insertion Rate 'INSR': INSR lies between 0 and 1 and indicates the percentage of the offspring (newly produced generation) that is re-inserted back into the old population. The operation of re-insertion is necessary in order to maintain the same number of individuals in successive generations, if a generation gap exists.

Number of frequency points 'L': This is the number of frequency points where a comparison between the exact and the test response is performed.

2.4.1 The methodology and pseudo GA code for FIR filters

The simple genetic algorithm used in the optimisation of FWL quantised coefficients of FIR filters is based on the standard techniques of generating the initial population of individuals followed by objective function calculation, ranking and crossover. No mutation operator was included in the algorithm as initial tests indicated no beneficial outcome of this operator for this optimisation problem. The GA used is a Matlab based toolbox designed and developed by Chipperfield et al [1993]. A number of standard functions are included in the toolbox that are indicated here by bold letters. For example **crtrp**(Nind, FieldDR) is a function that creates a random real-valued population of number of individuals 'Nind' with perturbation range of 'FieldDr' for each variable. The description of each stage of the GA process is as follows.

1) Generating initial population

The first step is to obtain the rounded integer valued coefficients from the design of the specified FIR filter that generates real-valued coefficients. The design option selected in this application is the 'remez' function of the Signal Processing toolbox of Matlab. This function is based on the Parks-McClellan optimal FIR filter design algorithm and is one of the most widely used FIR filter design technique. The filters designed using the 'remez' function are optimal in the minimax sense i.e. the maximum error between the desired and the actual frequency response of the filter is minimised. The integer programming optimised results of Kodek and Steiglitz [1981], with which the GA results are compared, also use the 'remez' function for the design of the FIR filters. It must be recognised that there is no loss of generality of the GA optimisation process if the initial design of the FIR filter is based on other standard design techniques. Some of these are the classical windowed technique used in the 'fir1' function or the weighted, integrated squared error minimisation used in the 'firls' function of the Matlab signal processing toolbox.

The design process for the FIR filters generates a set of coefficients of the form shown in Equation 2.1. The set of rounded integer valued coefficients are then derived using the Equation 2.3. This coefficient set forms the chromosome representation for GA optimisation. The population set of individuals is then generated using the **crtrp** function of the GA toolbox by randomly perturbing each rounded coefficient by +1, 0 or −1. This range of perturbation is obtained using the base value BASE=1. Increasing the base value to say 2 can extend this range and thus the search space. The random perturbation of coefficients will then be +2, +1, 0, -1 or −2. It must be noted that an appropriate choice of the base value depending on the filter length and the number of bits being used to represent the coefficients, is an area that needs further research. For this study, an initial trial of several different filters using base value of 1, 2 and 3 was conducted. The test results for a population size of 100 over 10 generations, consistently generated good results for base value of 1. An extensive range of search space could have been tested over larger population size and greater number of generations. However, the motivation for using GAs in this study was to test this optimisation process as a general framework against other methods and to draw a comparative measure.

2) Objective function evaluation

The main purpose of the optimisation process is to minimise the objective function with the specific aim of obtaining an approximated frequency response of the filter that is as close as possible to the desired response. The discrete search space for the example filters considered in this chapter can be calculated using the filter lengths and the base value used for coefficient perturbations. The filter length ranges from 15 to 35 and since these are linear phase filters then the actual number of coefficients that will be affected ranges from 8 to 18. The discrete search space for a base value of 1 is then $3^8 = 6561$for filter length 15 and $3^{18} = 387420489$ for filter length 35. This search space increases substantially when the base value is increased to 2. The

GA used for optimisation conducts 100 objective function evaluations initially followed by 80 evaluations over 10 generations. This makes a total of 900 evaluations using the base value of 1.

The objective function is calculated for 500 equally spaced frequency grid points. An example filter specification of a band select filter on a normalised frequency scale where Nyquist frequency = 1.0, is of the form:

Pass band range = 0 to 0.4 desired response = 1.0

Stop band range = 0.5 to 1.0 desired response = 0

The objective function is then evaluated using the following.

$$ObjV = \left\{ \sum_{i_p=0}^{p} | 1 - H_{i_p} |^2 + \sum_{i_s=s}^{L} | H_{i_s} |^2 \right\} + 10 \max \left\{ \max |1 - H_{i_p}|, \max | H_{i_s}| \right\} \qquad 2.24$$

Where H_{i_p} = magnitude response of GA optimised filter at frequency i_p in the pass band

H_{i_s} = magnitude response of GA optimised filter at frequency i_s in the stop band

L = number of frequency grid points (=500)

p = pass band cut-off point (=0.4 L)

s = stop band cut-off point (=0.5 L)

A combination of the summation of squared deviations and a weighted maximum deviation as seen in Equation 2.24 generated good overall frequency response that did not show the effects of skewing that was observed during initial trials when only the maximum deviation was used to optimise the objective function.

3) Fitness value and ranking

The Matlab based **ranking** function of the GA toolbox ranks the individuals according to their objective function values 'ObjV' and returns a column vector consisting of the corresponding fitness value 'FitnV' of the individuals. This function performs a linear ranking with a selective

pressure (SP) of 2 [Whitley, 1998]. The fitness value assigned to the individuals is calculated according the following formula given by Equation 1.1in Chapter 1.

4) Selection of individuals for breeding

The high-level function for selection of individuals from the population set and returning the selected individuals in a new population is performed by the 'select' function. The low-level selection function **sus** is called by the 'select' function. The **sus** function is based on a form of stochastic sampling method and is implemented by obtaining a cumulative sum of the fitness vector 'FitnV' and generating a set of equally spaced numbers between 0 and Σ(FitnV) [Baker, 1987]. The probability of an individual being selected is given by Equation 1.2 in Chapter 1.

5) Recombining individuals – crossover

The crossover function is also performed in two stages. The high-level function is **recombin** that calls the low-level function **recdis**. The **recdis** function is a discrete recombination function. The mating process is performed between pairs of rows. The **recdis** function first generates an internal mask table that determines which parents contribute which variables to the offspring. On the basis of the randomly generated mask table, the variable values are exchanged between the individuals and return a new population after mating.

6) Reinsert offspring into new population

The new population set generated after crossover is subjected to the objective function evaluation of each new individual. On the basis of their fitness, the offspring are selected for reinsertion into the new population. The objective function values are then copied according to the reinserted offspring and the GA loop is then repeated for the next generation.

A pseudo GA code for FIR filter optimisation and for the objective function are shown in Figures 2.3 and 2.4 respectively.

```
%  GA optimisation of FWL coefficients FIR digital filter

%  filter specifications and design

b = remez(ne,f,m); % filter design using remez algorithm

% GA parameters

[M,w]=freqz(b,1,L);

coefs=[round((2^n)*b(1:lc))]; % rounded coefficient values

% Build field descriptor FieldDR

% Initialise population
Chrom=round(crtrp(NIND,FieldDR));

% Evaluate objective function of initial population
ObjV=fir_obj(Chrom,lb,lc,M,L,n,R,fp,fs);

gen=0;   % counter

% Generational loop
while gen < MAXGEN

   %Assign fitness values to entire population
   FitnV = ranking(ObjV);

   %Select individuals for breeding
   SelCh=select('sus', Chrom, FitnV, GGAP);

   %Recombine individuals (crossover)
   SelCh=recombin('recdis',SelCh);

   %Evaluate offspring, call objective function
   ObjVSel=fir_obj(SelCh,lb,lc,M,L,n,R,fp,fs);

   %Reinsert offspring into population
   [Chrom ObjV]=reins(Chrom,SelCh,1,[1 INSR],ObjV,ObjVSel);

   %Increment counter
   gen=gen+1;
   [m,z]=min(ObjV);
   OBJ=ObjV(z,1);
  end
```

Figure 2.3 Pseudo GA code for finite word length coefficient optimisation of an FIR digital filter

```
%  Objective function
function f=fir_obj(Chrom,lb,lc,M,L,n,R,fp,fs);

[GA,w]=freqz(bb/(2^n),1,L);
GA=abs(GA);
GAp=GA(1:(fp*L));
GAs=GA((fs*L):L);

% objective function
f(i,1)=(sum((abs(1-GAp)).^2)+sum((abs(GAs)).^2))+10*max((max(abs(1-
GAp))),max(abs(GAs)));

% alternative objective function
% f(i,1)=max((max(abs(1-GAp))),max(abs(GAs)));
end;
```

Figure 2.4 Objective function pseudo code called by the main GA code for FIR digital filter

2.4.2 Example FIR filter for GA optimisation over all ω

This section deals with the GA optimisation of the direct form filter coefficients for the case of linear phase and of the arbitrary phase FIR filters over all ω. The maximum deviation $|E(\omega)|$ is determined over all ω in the range $0 \leq \omega \leq \pi$. The deviation $|E(\omega)|$ is derived from the error transfer function given by Equation 2.9 and represents the difference between the ideal filter response (using infinite precision coefficients) and the quantised coefficient filter response. The statistical bounds for the standard deviation of the error signals for linear phase and arbitrary phase FIR filters are given in Equations 2.13 and 2.14 respectively. It is conjectured here that the max $|E(\omega)|$ value for all ω will 'in all probability' be 3 times $\sigma_E(\omega)$. The max $|E(\omega)|$ deviation is then given by

$$\max |E(\omega)|_L = 3 \frac{Q}{2} \sqrt{\frac{2N-1}{3}} \qquad 2.25$$

for linear phase

and

$$\max |E(\omega)|_A = 3\frac{Q}{2}\sqrt{\frac{N}{3}} \qquad 2.26$$

for arbitrary phase

The example FIR filter used to test the validity of the error bounds of Equations 2.25 and 2.26 is designed using the Parks-McClellan (minimax) algorithm. The MATLAB function 'remez' is used to calculate the coefficients for this filter design.

The filter parameters used are

Filter length = 20 (total number of coefficients)

Frequency band edges = [0 0.4 0.5 1]

Desired magnitude response = [1 1 0 0]

Weighting function = 1 (both for pass and stop bands)

The GA parameters used are

Number of individuals (NIND) = 100

Maximum number of generations (MAXGEN) = 10

Generation gap (GGAP) = 0.8

Insertion rate (INSR) = 0.8

Peak variance of integer coefficients (BASE) = 1

Number of bits (B) = 5

Frequency axis number of points = 500

The error objective function used for optimisation is given by Equation 2.24

Case 1 – Linear Phase FIR filter

The magnitude response of the rounded coefficient value and GA optimised filter for a 5-bit coefficient representation is shown in Figure 2.4. The phase response is not included here, as

this will be linear because of the symmetrical coefficients used for the optimisation process. Figure 2.5 shows the magnitude error max $|E(\omega)|_L$ for the rounded coefficients, the GA optimised coefficients and the bounded value given by Equation 2.25.

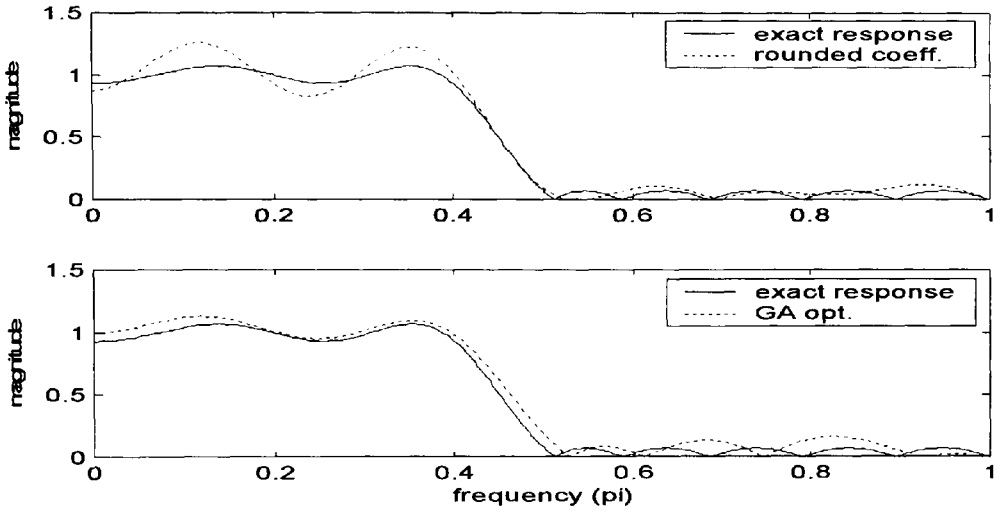

Figure 2.5 Magnitude response of simply rounded and GA optimised coefficient filter.

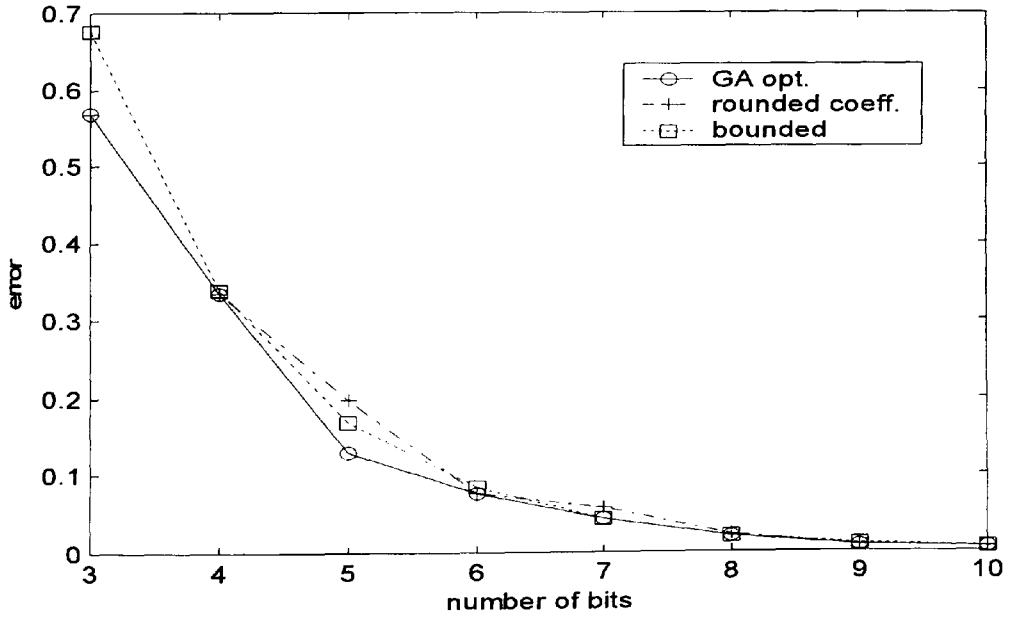

Figure 2.6 Comparison of error magnitudes max $|E(\omega)|_L$

The rounded and the GA optimised coefficients are obtained respectively as

h(rnd) = 0 -1 -1 0 1 0 -1 -1 3 7 7 3 -1 -1 0 1 0 -1 -1 0

$h(GA\text{-}op)_L$ = 0 0 -1 0 1 0 -1 -1 3 7 7 3 -1 -1 0 1 0 -1 0 0

Case 2 – Arbitrary Phase FIR filter

In general, non-symmetrical coefficient, i.e. arbitrary phase FIR filters are not used in practical applications due to the requirement for doubling the memory space to store the relevant coefficients and also the loss of phase linearity. However, there may be useful application of such filters for low-bit low-order implementation leading to computationally efficient, low power systems. For this reason, a GA optimisation code was developed with minor amendments to the code of Appendix B using the same filter characteristics and GA parameters as shown above in this section. The magnitude response of the rounded coefficient filter and of the GA optimised filter is shown in Figure 2.6 that also includes the phase response of the GA optimised filter. The number of bits used for the coefficients of this filter is 5. Figure 2.7 shows the magnitude error max $|E(\omega)|_A$ for the rounded coefficients, the GA optimised coefficients and the bounded value given by Equation 2.26.

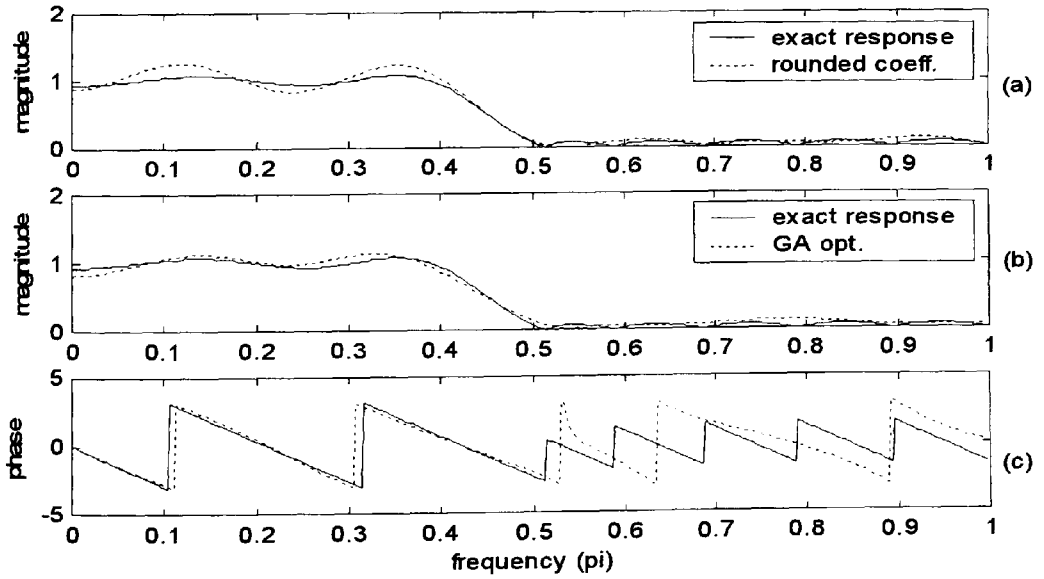

Figure 2.7 Magnitude response of simply rounded filter (a), GA optimised filter (b) and phase response of GA optimised filter (c).

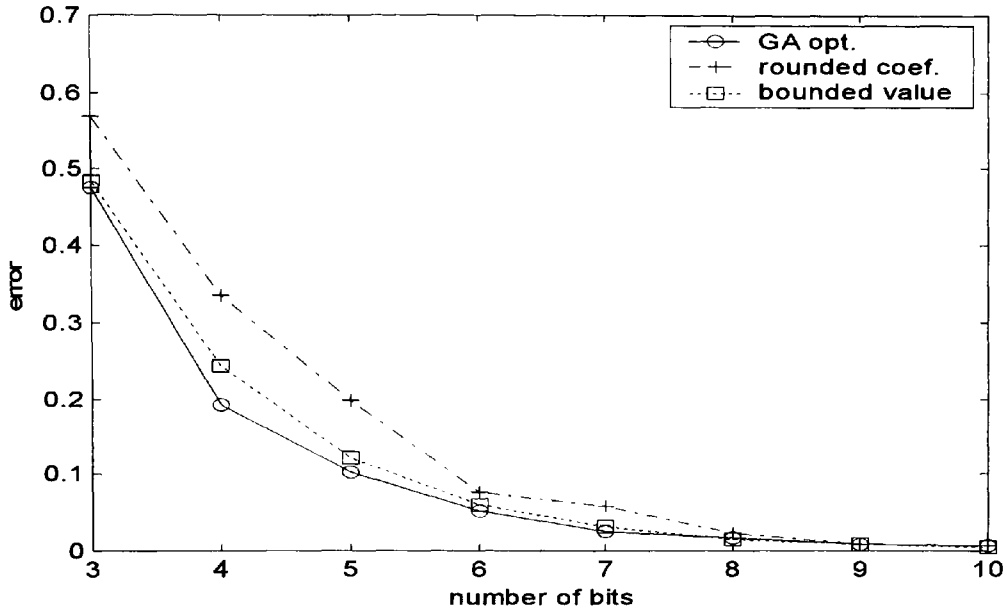

Figure 2.8 Comparison of error magnitudes max $|E(\omega)|_A$

The rounded and the GA optimised coefficients are obtained respectively as

h(rnd) = 0 -1 -1 0 1 0 -1 -1 3 7 7 3 -I -1 0 1 0 -1 -1 0

$h(GA\text{-}op)_A$ = 0 -1 -1 0 1 1 -1 -1 3 6 7 3 -1 -2 -1 0 0 0 0 0

The graphs shown in Figures 2.5 and 2.7 demonstrate a good parity between the GA optimised results and the bounded value of Equations 2.25 and 2.26 thus suggesting that the original conjecture of max $|E(\omega)|$ = 3 times $\sigma_E(\omega)$ is closely valid.

2.4.3 GA optimisation of band select FIR filters

This section deals with the case of band select filters for which the desired response is specified over the selected pass and stop bands. The desired function $D(\omega)$ consists of a number of disjointed frequency bands $\Omega_k \subset [0,\pi]$, where k=1,...,M such that for each k, $D(\omega)$ is to be

approximated to within a specified error bound $\delta(k)$ for all $\omega \in \Omega_k$. In order to conduct a comparative study, the 10 filter examples used by Kodek and Steiglitz [1981] for coefficient optimisation based on integer programming method are also used here. The 10 filters are divided into 4 sets of filters as shown in Table 2.2. The filter coefficients for the 10 filters are shown in Table 2.3. The representation such as A15/5 denotes A-range filter of length 15 using 5 bit coefficients. The integer programming method optimised coefficients have been taken from [Kodek and Steiglitz, 1981].

Table 2.2

Sets of Filter Specifications

Filter	**Pass-band**	**Stop-band**	**Pass-band**
A: range	0 to 0.4	0.5 to 1.0	
Weighting:	1	1	
Desired value:	1	0	
B: range	0 to 0.4	0.5 to 1.0	
Weighting:	1	10	
Desired value:	1	0	
C: range	0 to 0.24	0.4 to 0.68	0.84 to 1.0
Weighting:	1	1	1
Desired value:	1	0	1
D: range	1 to 0.24	0.4 to 0.68	0.84 to 1.0
Weighting:	1	10	1
Desired value:	1	0	1

Table 2.4 shows the results for the maximum error deviation to the desired response for all $\omega \in \Omega_k$ and Table 2.5 shows the results for the total summation error relative to the desired response for all $\omega \in \Omega_k$. The bounded value used in Table 2.4 is obtained using Equation 2.20. A comparison with the integer programming (IP) method clearly shows a distinct improvement for

the case of GA optimised filters both for the maximum error deviation and for the summation error. It is also observed that the bounded value of Equation 2.20 is consistent with the maximum error deviation obtained using the GA optimised filters. The GA optimised filters have generated slightly lower maximum error deviation values as compared to the value of bounded error for all but two of the ten filters namely, the A25/5 and the C15/5 filters. On the other hand, the IP optimised filters have better performance compared to the bounded value in just two of the ten filters.

The reason for comparison of the total summation error, as shown by results in Table 2.5, is useful as this generates a more distinct semblance with the desired response of the filter without the possibility of skewing the overall frequency response. Again, it can be seen that the GA optimised filters have significantly outperformed those using the IP optimised filters in most instances. In comparison to the IP optimised filters, the GA optimised results have marginally under performed in one case namely the A15/5 filter, for two other filters namely A25/5 and C25/5, the results are identical and for the remaining seven filters, the GA optimised results are distinctly superior.

Some example filter responses for filters B25/7 and C25/5 are shown in Figures 2.8, and 2.10 respectively. It is observed that while the rounded response follows the exact response as is to be expected, the GA optimised response follows the requirement of the desired response which is 1 in the pass band for the filter B25/7. It is also observed that the optimised filters, as shown in Figure 2.10 have significant deviation from the exact response in the transition region. This deviation may not be critical for a required design however, it is sufficiently significant for consideration of a specific filter response realisation. Figures 2.11 and 2.12 show a comparison of maximum error magnitudes against number of bits B for filters A15 and B25 respectively.

Table 2.3

The filter coefficients of the 10 filters. 'GA-op' denotes GA optimised, 'Rnd' denotes rounded and 'IP-op' denotes the integer programming method optimised. Only half the coefficients are given due to symmetrical property.

Filter 1: A15/5																	
GA-op	7	5	0	-1	-1	1	1	0									
Rnd	7	5	1	-1	-1	1	1	0									
IP-op	7	5	1	-2	0	1	1	0									
Filter 2: A25/5																	
GA-op	7	5	1	-1	-1	1	1	0	-1	0	0	0	0				
Rnd	7	5	1	-1	-1	1	1	0	0	0	0	0	0				
IP-op	7	5	1	-1	-1	1	1	0	-1	0	0	0	0				
Filter 3: B15/7																	
GA-op	29	20	3	-6	-3	2	5	2									
Rnd	28	20	4	-6	-3	3	6	3									
IP-op	28	20	3	-7	-5	2	5	3									
Filter 4: B25/7																	
GA-op	28	20	3	-6	-3	2	3	-1	-2	0	1	1	0				
Rnd	28	20	4	-6	-3	2	3	-1	-2	0	2	2	1				
IP-op	27	19	3	-6	-3	3	3	-1	-3	-1	2	2	1				
Filter 5: B35/7																	
GA-op	28	20	4	-6	-3	2	3	-1	-2	0	1	0	-1	-1	1	1	0	0
Rnd	28	20	4	-6	-3	2	3	-1	-2	0	1	1	-1	-1	1	1	1	0
IP-op	28	20	4	-5	-3	2	2	-1	-2	0	1	0	-1	0	2	2	1	0
Filter 6: C15/5																	
GA-op	8	1	5	-1	0	-1	-1	1									
Rnd	9	1	5	-1	-1	0	-1	1									
IP-op	9	1	5	-1	0	-1	-1	1									
Filter 7: C25/5																	
GA-op	9	0	5	0	-1	0	-1	0	1	0	0	0	-1				
Rnd	9	1	5	-1	-1	0	-1	1	0	0	0	0	0				
IP-op	9	0	5	0	-1	0	-1	0	1	0	0	0	-1				
Filter 8: D15/7																	
GA-op	34	4	20	-4	0	-2	-3	2									
Rnd	35	4	21	-3	1	-3	-2	1									
IP-op	34	4	20	-4	0	-4	-3	0									
Filter 9: D25/7																	
GA-op	34	3	19	-4	-2	-2	-4	3	1	1	1	-2	0				
Rnd	34	3	19	-4	-2	-2	-4	3	1	1	1	-1	0				
IP-op	34	3	19	-4	-1	-2	-2	3	2	1	1	-1	0				
Filter 10: D35/7																	
GA-op	35	3	19	-4	-2	-2	-4	3	1	1	1	-2	0	0	0	1	0	0
Rnd	34	3	19	-4	-2	-2	-4	3	1	1	1	-2	0	0	0	1	0	0
IP-op	34	3	19	-4	-2	-2	-4	3	1	1	1	-2	0	-1	0	0	0	0

Table 2.4
Maximum error deviation relative to the desired response for all $\omega \in \Omega_k$

filter	GA-op	Rounded	IP-op	exact	Bounded value (Equation 2.20)
A15/5	0.1978	0.2309	0.2002	0.1324	0.2296
A25/5	0.1873	0.2309	0.1873	0.0508	0.1771
B15/7	0.2315	0.2813	0.3273	0.2797	0.3040
B25/7	0.0993	0.1251	0.2157	0.1231	0.1547
B35/7	0.0637	0.0869	0.1865	0.0528	0.0903
C15/5	0.1672	0.1873	0.1667	0.0596	0.1568
C25/5	0.1265	0.1873	0.1265	0.0173	0.1436
D15/7	0.1483	0.2143	0.2542	0.2006	0.2249
D25/7	0.0428	0.0651	0.1306	0.0570	0.0886
D35/7	0.0425	0.0558	0.0668	0.0152	0.0526

Table 2.5
Total summation error relative to the desired response for all $\omega \in \Omega_k$

filter	GA-op	Rounded	IP-op	exact
A15/5	6.1843	3.5146	5.9662	3.1958
A25/5	3.8005	3.5146	3.8005	0.3576
B15/7	4.5658	7.3014	6.8917	7.7252
B25/7	0.5421	1.6740	2.7130	1.5205
B35/7	0.5158	0.7558	2.0358	0.2805
C15/5	1.7789	2.0507	2.9096	0.3663
C25/5	1.4342	2.0507	1.4342	0.0298
D15/7	1.0987	3.6671	2.8557	4.3325
D25/7	0.2187	0.2359	1.2669	0.2327
D35/7	0.1556	0.1360	0.3666	0.0112

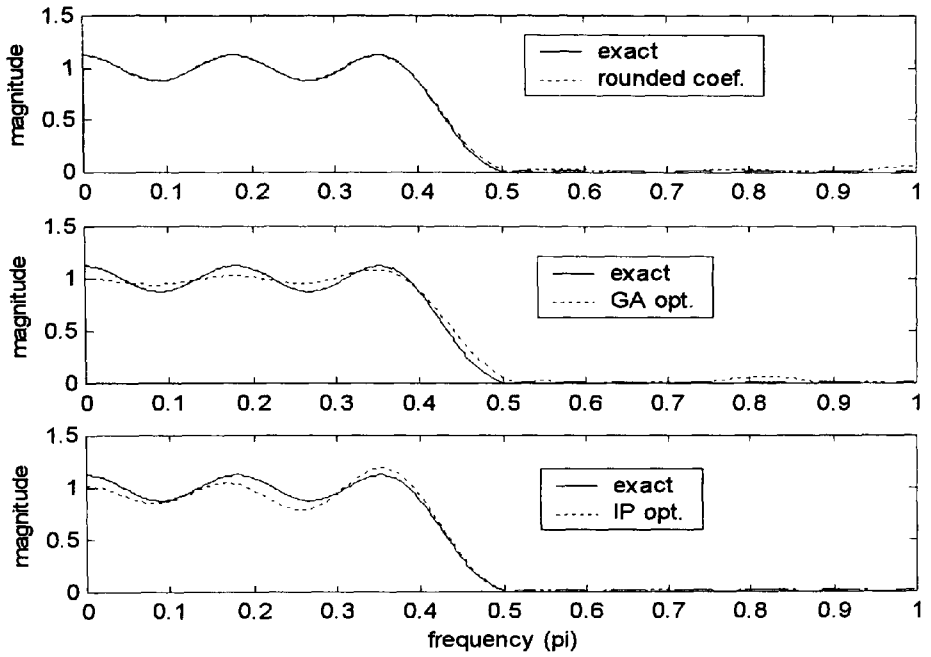

Figure 2.9 Magnitude response of simply rounded, GA optimised and IP optimised coefficients for the case of filter B25/7.

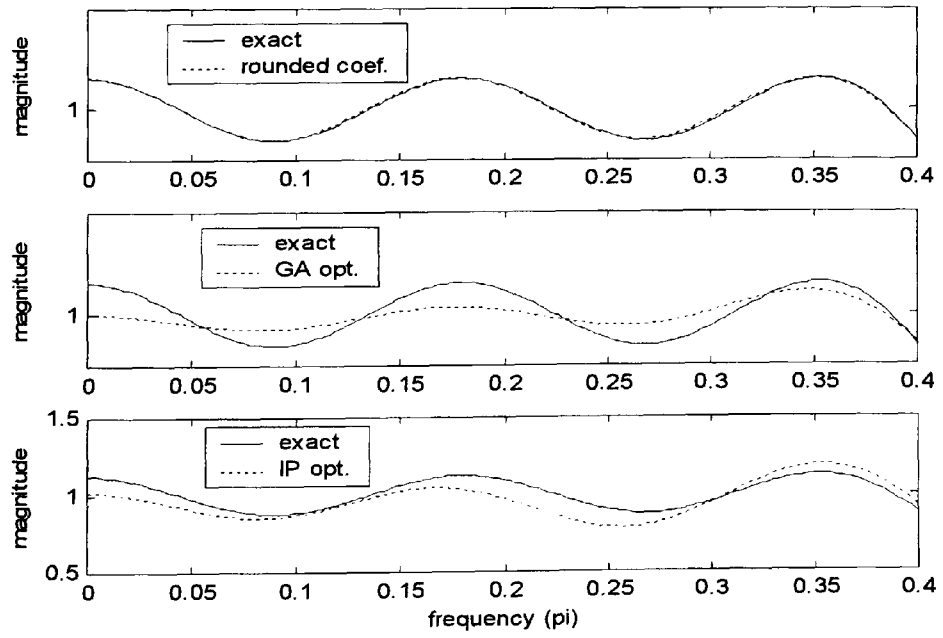

Figure 2.10 Magnified response of simply rounded, GA optimised and IP optimised coefficients for the case of filter B25/7.

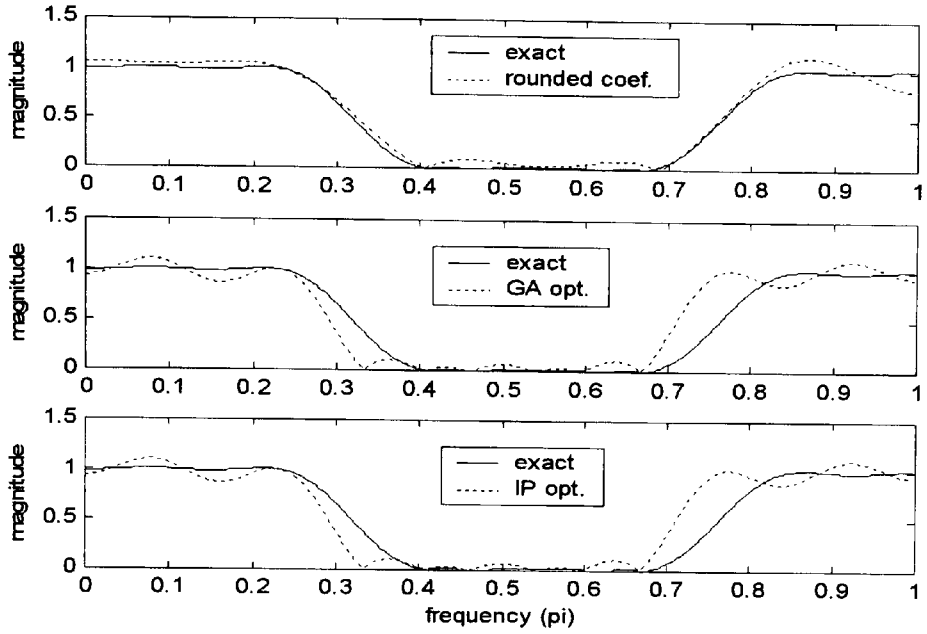

Figure 2.11 Magnitude response of simply rounded, GA optimised and IP optimised coefficients for the case of filter C25/5.

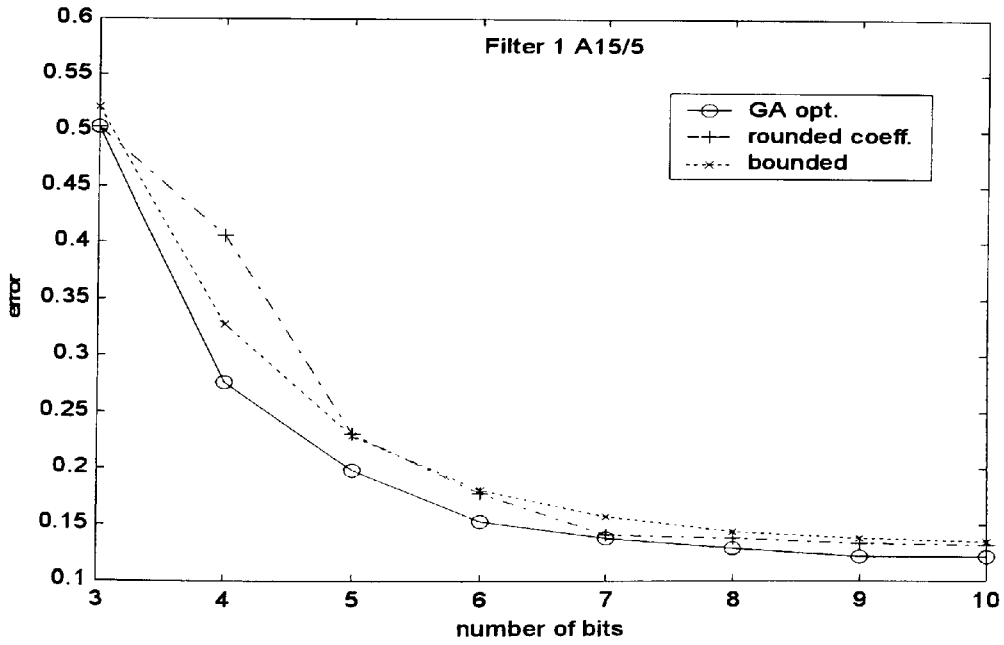

Figure 2.12 Comparison of error magnitudes against number of bits B for filter A15.

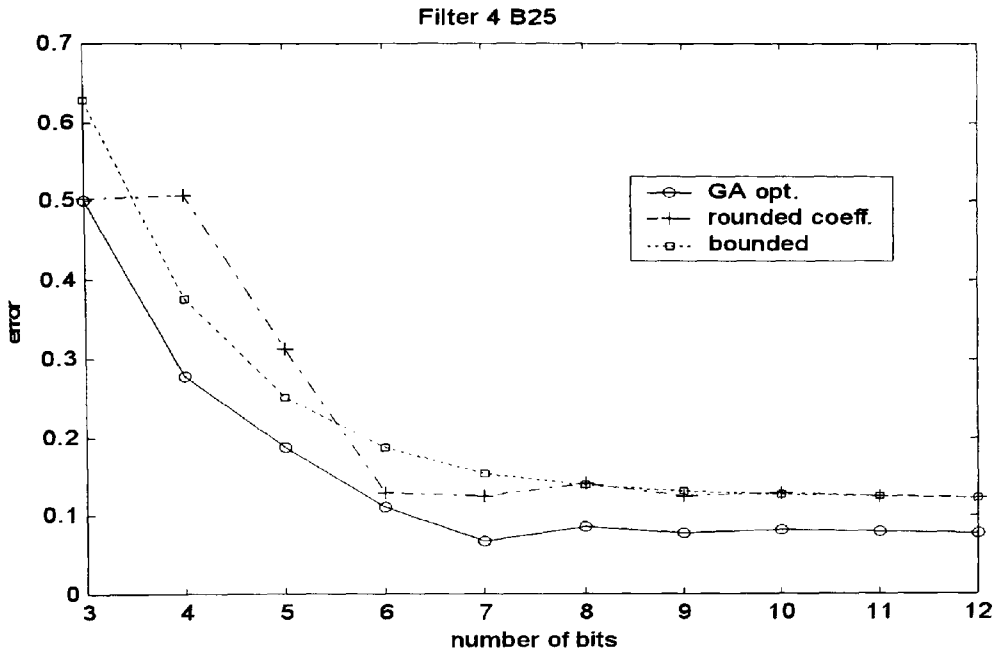

Figure 2.13 Comparison of error magnitudes against number of bits B for filter B25.

2.5 Simple hill climber techniques and exhaustive search

To test the robustness and accuracy of the GA optimised results, the methods of simple hill climber algorithms such as the steepest ascent (SAHC) and the nearest ascent (NAHC) were applied to a selection of filters shown in Table 2.4. Random sampling tests for the search space as used for the GA optimisation was also conducted. Furthermore, for a small selection of low order filters, an exhaustive search was conducted over a matching search space. The hill climber algorithms for this search were based on the standard techniques used for binary strings [Mitchell, 1996] and adapted for the case of integer valued numbers representing the FIR filter coefficients. It must be recognised that the integer valued rounded coefficients of FIR filters are derived from real valued coefficients that are represented by a finite number of bits. The hill climber technique in this context for real-valued variable is synonymous with the integer valued variable. The starting 'seed' individual of an integer valued coefficient set is generated by randomly perturbing the rounded coefficients by +1, 0 or −1. The flow chart shown in Figure

2.14 describes the hill climber algorithm used for this application. In order to maintain parity with the GA optimisation, approximately the same number of objective function evaluations were performed for the hill climber methods. The hill climber performed a maximum of 90 objective function evaluations i.e. 9 evaluations (for filter length 15) for each loop running a maximum of 10 times. 10 runs of each algorithm, each starting with a different randomly generated seed thus generates a maximum of 900 evaluations; the same number as the GA evaluations.

The SAHC generates new neighbours by systematically mutating each coefficient randomly by +1, 0 or -1 moving from left to right. For example, if the starting randomly mutated 'seed' coefficient string is -4, 2, 1, 0 then the neighbours can be -5, 2, 1,0; -4, 3, 1, 0; -4, 2, 0, 0 and -4, 2, 1, 1. For the NAHC algorithm the neighbours are derived by mutating each of the coefficients from left to right successively while keeping the previously mutated coefficients. For example, if the starting coefficient string is -4, 2, 1,0 then the neighbours can be -5, 2, 1, 0; -5, 3, 1, 0; -5, 3, 0, 0 and -5, 3, 0, -1. An important observation for the application of the above described hill climber algorithms is that the search space for optimisation can extend beyond the range of +1 or -1 of the rounded values for each coefficient. This outcome is implicit in the evolutionary nature of the algorithms since mutation of the coefficient value occurs for each iteration. In this respect, there is a subtle difference when compared with the GA optimisation because the search space for GA is confined to +1 and -1 of the rounded coefficient values for the results obtained in this study. The hill climbers are thus subjected to a wider search space that may or may not be advantageous to the optimisation process. There is a possibility of obtaining a superior solution when compared to the GA method, however there is also a danger for the search to move towards areas of inferior or local minima solutions. The results of SAHC, NAHC, the random sampling and exhaustive search for a selection of the FIR filters are shown in Table 2.6 and the filter coefficients are shown in Table 2.7. The results shown with an

asterisk (*) are the ones for which the search space has deviated greater than +1 or -1 of the rounded coefficient values. Note also that the exhaustive search was confined to deviation of +1, 0 or −1 of the rounded coefficients.

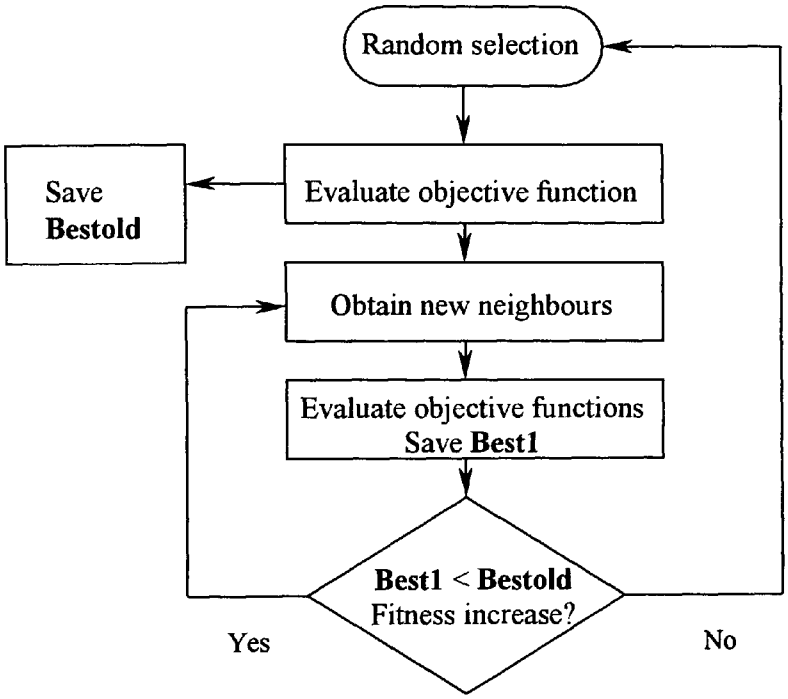

Figure 2.14 A flow chart for the simple hill climber algorithm.

Table 2.6
Maximum error deviation relative to the desired response for all $\omega \in \Omega_k$

Filter	Exh. search	Random	SAHC	NAHC	GA
A15/5	0.1978	0.1978	0.1978	0.3536	0.1978
A25/5	None	0.3051	0.1873	0.2517	0.1873
B15/7	0.2208	0.2322	0.1418*	0.1972*	0.2315
B25/7	None	0.1088	0.0678*	0.0740*	0.0993
C15/5	0.1667	0.1875	0.1667	0.2796	0.1672
C25/5	None	0.2468	0.1576	0.2623	0.1265

Exh. = Exhaustive
SAHC=steepest ascent hill climber
NAHC=nearest ascent hill climber
* indicates search space exceeded +1 and/or −1 of rounded coefficient values

Table 2.7

The filter coefficients of selected filters. Only half the coefficients are given due to symmetrical property.

Exh. Sch= Exhaustive
Rand=Random sample
SAHC=steepest ascent hill climber
NAHC=nearest ascent hill climber
* indicates search space exceeded +1 and/or −1 of rounded coefficient values

Filter 1: A15/5													
Exh. Sch	7	5	0	-1	-1	1	1	0					
Rand	7	5	0	-1	-1	1	1	0					
SAHC	7	5	0	-1	-1	1	1	0					
NAHC	8	5	0	-2	-1	1	1	-2					
Filter 2: A25/5													
Exh. Sch	none												
Rand	7	5	1	-2	-1	0	0	1	-1	0	0	0	0
SAHC	7	5	1	-1	-1	1	1	0	-1	0	0	0	0
NAHC	7	5	1	-2	-1	1	1	0	-1	0	0	0	0
Filter 3: B15/7													
Exh. Sch	28	20	3	-7	-2	2	5	2					
Rand	29	19	3	-7	-2	2	5	2					
SAHC	28	20	3	-6	-3	3	5	-1*					
NAHC	28	21	2	-5	-3	4	3*	1*					
Filter 4: B25/7													
Exh. Sch	none												
Rand	28	19	4	-6	-4	2	3	-2	-2	0	1	2	0
SAHC	28	20	3	-6	-3	2	2	-1	-2	0	2	1	-1*
NAHC	28	20	3	-6	-3	2	2	-1	-2	1	2	0*	-1*
Filter 5 C15/5													
Exh. Sch	9	1	5	-1	0	-1	-1	1					
Rand	9	1	5	-2	0	-1	-1	1					
SAHC	9	1	5	-1	0	-1	-1	1					
NAHC	9	1	5	-1	0	0	-1	-1					
Filter 6 C25/5													
Exh. Sch	none												
Rand	9	0	5	0	-1	1	-1	0	1	0	0	0	0
SAHC	9	1	5	-1	-1	0	-1	1	1	0	0	-1	0
NAHC	10	1	5	0	0	-1	-1	0	1	0	0	0	0

2.6 Discussion of results

Section 2.4.2 covers the results of the maximum deviation between the exact filter magnitude response and the FWL coefficient results. Established maximum deviation theoretical bounds calculated using the number of bits to represent the filter coefficients and the filter length are used to draw a comparison with the deviation obtained by simply rounded valued coefficients and the GA optimised FWL coefficients. The case of a linear phase and of arbitrary phase FIR filters is considered. The graphical results of the magnitude response of a typical low pass FIR filter of length 20 for the linear and arbitrary phase responses are shown in Figures 2.5 and 2.7 respectively. The maximum deviation error between the exact and the approximate magnitude responses for a number of bits ranging from 3 to 10 representing the exact coefficient values is shown in Figures 2.6 and 2.8 for the linear and arbitrary phase FIR filters respectively. It is clearly seen that the GA optimised results generate the minimum deviation error and is closely followed by the theoretically predicated error bounds given by Equations 2.25 and 2.26. The simply rounded coefficients generate the worst error. Furthermore, the assumption made in section 2.4.2 that the maximum deviation is likely to be three times $\sigma_E(\omega)$ holds well as is evident from the results shown in Figures 2.5 and 2.7.

For the case of band select FIR filters covered in section 2.4.3, the choice of filters shown in Table 2.2 were taken directly from [Kodek and Steiglitz, 1981]. These filter coefficients were optimised using the GA code developed for this study (see Appendix B) and then compared with the integer programming method optimised coefficient results as listed by Kodek and Steiglitz [1981]. Table 2.3 lists the coefficient values of the ten band select filters for the rounded values, the integer programming method optimised values and the GA optimised values. Table 2.4 lists the values for maximum deviation relative to the desired response and Table 2.5 lists the total summation error relative to the desired response. These values clearly

show significant improvement of the GA optimised results for most of the example band select filters when compared to the integer programming method optimised results. This is evidence of the efficacy of the GA technique as an optimisation tool in the specific application of FWL constraints on the infinite precision coefficients of the FIR digital filters.

Further tests were conducted on a selection of FIR filters using the simple hill climber techniques, random sampling and exhaustive search. The results of these tests are shown in Table 2.6. Once again, the GA optimised results are seen to be consistently good. However, for some filters such as the B15/7 and B25/7, the hill climber methods have generated superior results. This is significant since the search space for these algorithms can intrinsically extend beyond the +1 and/or −1 of the rounded coefficient values. The GA search space, however, is restricted to +1 or −1 of the rounded coefficients. This offers credibility to the simple hill climber technique and complements the GA optimisation to search for superior solutions for the application considered here.

2.7 Summary of Chapter 2

The specific problem of finite word length coefficients in the realisation of FIR filters has been considered here. The purposeful aim is to use the procedures of genetic algorithms to optimise the frequency response in comparison to the exact filter response and to the desired response for the case of band select filters. The GA programme used in this application is explained and a specific code is developed to seek optimal results on the basis of the minimisation of a predefined error objective function. Quantifiable metrics for comparison purposes are defined on the basis of the maximum error bound $|E(\omega)|$ for all ω both for the case of linear phase and arbitrary phase FIR filters. Comparison is drawn between the simply rounded coefficient results and those obtained using the GA optimised coefficients. In both cases of FIR filters, it is

observed that good near optimal results of filter coefficients are obtained using the GA method. Furthermore, the bounded values given in Equations 2.13 and 2.14 show a good parity with the GA optimised results as seen in Figures 2.6 and 2.8 for the case of linear phase and of the arbitrary phase FIR filters respectively.

For the case of band select filters, a comparison is drawn with the results taken from [Kodek and Steiglitz, 1981] for ten specified FIR linear phase filters that are optimised using the integer programming method. The GA optimised results show a distinct improvement over the integer programming method of optimisation both for maximum error and for total summation error within the specified range of band selected frequencies. The results of maximum error against number of bits for the case of example filters as seen in Figures 2.12 and 2.13 show consistently lower values obtained by GA optimised results in comparison to rounded coefficients response or the bounded values of Equation 2.20.

The general conclusion of this part of the study leads to the observation that FIR filters are fairly accurate in their frequency response realisation using quantised rounded valued coefficients. A distinct measure of improvement is achievable by using GA optimisation especially for low number of bits (see Figures 2.5, 2.6, 2.12 and 2.13). The bounded error results shown in Figures 2.6, 2.8, 2.12 and 2.13 show a good correspondence between the statistically calculated bounds and the results obtained for GA optimised filters. For completeness, a study of non-symmetrical FIR filters has also been conducted using the same metrics for comparison as mentioned above. The results of Tables 2.4 and 2.5 clearly indicate a substantial improvement of GA optimised frequency response over the simply rounded response and also over the optimised results using the integer programming method. The GA code running on a 600MHz pentium-3 computer with the parameters given in Section 2.4.2 above completed the optimisation in approximately 30 seconds for each filter.

The major contributions of this chapter are the following.

- A real integer-valued genetic algorithm code has been developed for the optimisation of finite word length constrained coefficients of FIR digital filters. This code also incorporates the option for preserving the zero-valued coefficients that occur during the original high-precision design of the FIR digital filters. This option allows for preserving the memory space allocation for the coefficients. The new GA optimised results are significantly superior when compared with the integer programming method optimised results taken from [Kodek and Steiglitz, 1981]. The comparative results are shown in Tables 2.4 and 2.5 and the new coefficient values are listed in Table 2.3. Other comparative results for a selection of filters using the simple hill climber techniques, random sampling and exhaustive search are shown in Table 2.6 and the new coefficient values are listed in Table 2.7.

- The GA optimised results for FIR filters demonstrate the assertion that the maximum deviation derived using statistical methods [Chan and Rabiner, 1973] as given by Equations 2.25 and 2.26 holds well as is evident from the GA optimised results seen in Figures 2.6 and 2.8.

The GA code developed and applied for the optimisation of FWL coefficients FIR digital filters is extended for the case of infinite impulse response (IIR) digital filters. The discussion and FWL coefficient optimisation of IIR filters using genetic algorithms form the basis for study that is covered in the next chapter.

Chapter 3: Finite word length optimisation of IIR filters

Overview of Chapter 3: This Chapter starts by highlighting the specific problems of finite word length (FWL) coefficient constraints for the case of infinite impulse response (IIR) digital filters. Due to the feedback nature of recursive filters, stability issues are an important factor in their design and are discussed in some detail. Some previously reported work on the optimisation of FWL coefficients for IIR filters is also discussed. Extensive range of filter types and structures of IIR filters and their optimisation using genetic algorithms is investigated and reported. Finally, comparative tests were conducted using the simple hill climber optimisation techniques for a selection of filters.

3.1 Introduction

The investigation of finite word length constraints on FIR filter coefficients that was covered in Chapter 2 is followed with a similar study in this chapter for the case of IIR filters. The exact design and analysis of IIR filters is normally based in terms of linear systems. However, when finite word length effects of quantisation error and overflow are considered then the system becomes non-linear and it is this that causes difficulties in the analysis of fixed point filter implementation. For recursive filter structures the problems of the effects of finite word length become more severe when compared to the non-recursive filters. In an extreme situation and especially for narrow band filters where the poles of the filter are fairly close to the unit circle, the finite word length coefficients of IIR filter may generate positive feedback and thus become unstable. The finite word length realisation of recursive IIR filters due to fixed-point hardware suffer from the same error effects as for non-recursive FIR filters such as; input signal quantisation due to analogue to digital conversion, coefficient quantisation, overflow errors and product round-off errors. In addition to these errors, two additional problems are caused by the

feedback nature of the recursive filter. These are; firstly, small-scale limit cycles that are oscillations caused by quantisation non-linearity in the feedback loop and secondly, large scale limit cycles caused by fixed-point arithmetic non-linear overflow in the feedback loop. Of these two problems, the first error is usually low and is easily tolerated but the second error can lead to large-amplitude sustained oscillations over the complete dynamic range of the recursive filters and so must be prevented. For the purpose of analysing finite word length effects in recursive filters, such filter errors can be classed in four categories, these are; filter coefficient errors, quantisation noise and overflow errors and the two limit cycles. The impact of these errors is significantly influenced by the structure of the recursive filters used for their realisation.

The commonly used structures are; direct, parallel and cascade forms. It is well established that a cascade of 2nd order sections is much less sensitive to coefficient quantisation effects and its impact on stability of the filter, especially when compared to direct form implementation [Kaiser, 1965], [Parks and Burrus, 1987]. Proper ordering and matching of poles and zeros in each section of the cascade and their scaling further ensures reduced coefficient sensitivity [Oppenheim and Shaffer, 1989], [Jackson, 1989]. Other structures that can be used for IIR filter realisation are: state-variable [Roberts and Mullis, 1987], lattice [Gray and Markel, 1973] and the wave digital filter [Fettweis, 1974]. Generally, as the filter structure becomes more complex then a larger part of the process of filtering is carried by the structure itself placing less load on the coefficients. Thus more complex structures such as lattice and wave digital filters, are capable of operating to a required response, with greatly reduced coefficient word lengths. However, in terms of coefficient sensitivity, the 2^{nd} order cascade has been shown to perform well even when compared to parallel and wave digital filter structure for a number of designs [Dempster and Macleod, 1994]. Hence, in this study, 2^{nd} order cascades are used due to their simplicity of filtering algorithms and ease of performing stability tests.

3.2 Optimisation issues in IIR filter design and stability

In a recursive digital filter the output is a linear combination of past and present inputs and past outputs. The difference Equation representation is of the form

$$y(n) = \sum_{m=0}^{M} b_m x(n-m) - \sum_{m=1}^{N} a_m y(n-m) \qquad 3.1$$

and the transfer function is

$$H(z) = \frac{\sum_{m=0}^{M} b_m z^{-m}}{1 + \sum_{m=1}^{N} a_m z^{-m}} \qquad 3.2$$

The coefficients b_m and a_m of the filter transfer function H(z) of Equation 3.2 are obtained in a high precision form through the initial design stage of the IIR filter such that the stipulated filter specifications are satisfied. For real-time realisation of such filters using fixed-point devices, then errors arise in a number of ways that could degrade the performance of the filter and in extreme situations make an otherwise stable filter to become unstable. The main sources of errors in the IIR filter realisation due to finite word length effects are the following.

- The analogue to digital conversion (ADC) quantisation noise that results from representing the input analogue signal in a quantised discrete form.
- Filter coefficient quantisation error that is caused by representing the high precision coefficients by a finite number of bits.
- Overflow errors that are due to the finite-precision arithmetic operations of addition, multiplication and storage.

In addition to the above, two other sources of errors are caused by the feedback nature of the recursive IIR filters, these are:

- Small-scale limit cycles that are caused by the quantisation non-linearity in the feedback loop. These are generally of small amplitude and can be tolerated.
- Large-scale limit cycles that are oscillations caused by the overflow of arithmetic operations in the feedback loop. These overflow amplitudes can cover the complete dynamic range of the filter and so must be strictly prevented.

Appropriate scaling of the inputs to the adders such that the outputs are kept low can prevent the large-scale limit cycles. This can lead to a reduction of the signal-to-noise ratio (SNR) of the system. It is thus important to select an appropriate scaling factor for a given structure to prevent overflow while at the same time preserving the best possible SNR. There are three commonly used methods for determining a suitable scaling factor and are referred to as the L_1, L_2 and L_∞ norms [Parks and Burrus, 1987], [Ifeachor and Jervis, 1993].

Of the above listed sources of error, the ADC quantisation noise error is inevitable and is dependent on the number of bits used to represent the input analogue signal in a digital form. Increasing the number of bits used for ADC can only reduce this form of noise. The error due to FWL quantised coefficient representation of the filter transfer function forms the focus of study in this work and its optimisation is considered here in some detail. Finally, the round-off noise, pairing and ordering of poles and zeros and scaling are important issues that are considered mainly in the design stage of the IIR filter. No effort was devoted to the FWL optimisation of these issues either individually or simultaneously. However, it must be emphasised that a combination of effective initial design taking account of the scaling factor and the ordering of poles and zeros gives a good starting point for the FWL coefficient optimised realisation of the IIR filter. However, final checks may be needed to confirm the validity of the optimised design. A combined optimisation of multiple parameters for the FWL realisation of IIR filters is an area that has yet to be fully investigated.

The major effect of filter coefficient quantisation into a finite number of bits is that the infinite precision transfer function H(z) is no longer representative since the poles and zeros are shifted in the z-plane. The new transfer function is then an approximation of the original and in some instances, especially for narrow band filters, the poles that are close to but inside the unit circle, may have shifted sufficiently to place their new position outside the unit circle. This condition will lead to the filter becoming unstable. The finite word-length coefficient optimisation for recursive IIR filters must, therefore, consider both the satisfactory frequency response and the stability of the filter. The issue concerning coefficient sensitivity and filter stability has been analysed by considering the extent to which the pole positions change as a result of changes in the coefficient a_m [Parks and Burrus, 1987 pp.234-236], [Mitra, 1998 pp.578-582]. Some of the important deductions of the analysis are

- The filter is most sensitive to the last coefficient a_N.
- Moving the poles closer to the unit circle increases the sensitivity of the pole location to the variation of a coefficient.
- Coefficient sensitivity increases when the poles are close together.
- For high order filters, the poles are normally clustered together in the pass band, so in order to reduce sensitivity a cascade structure of several lower order sections is recommended. This leads to the preference for using a cascade of 2^{nd} order section.

By factoring the rational transfer function of Equation 3.2, the cascade 2^{nd} order transfer function can be written as

$$H(z) = \prod_{k=1}^{L} \frac{b_{0k} + b_{1k} z^{-1} + b_{2k} z^{-2}}{a_{0k} + a_{1k} z^{-1} + a_{2k} z^{-2}} \qquad 3.3$$

where L is the number of cascade sections.

There are many different cascade structures possible depending on the ordering of $H_k(z)$ blocks and also on different pairings of the poles and zeros. This flexibility of structural leniency allows for a choice of filter realisations that may be used to reduce coefficient quantisation dependency. For reduced coefficient sensitivity, a specific 2^{nd} order cascade design procedure is described in [Jackson, 1989]. Some important considerations in the ordering of the cascade sections are

- Match the poles closest to the z-plane unit circle with the zeros closest to those poles.
- Match the remaining poles to zeros similarly, moving towards the origin of the unit circle.
- The section with poles closest to the unit circle is ordered as the last section of the cascade preceded by other pole/zero pairing sections according to the distance of the poles from the unit circle.

The condition of (iii) above is based on the assumption that the following scaling of the sections in the cascade is applied i.e.

$$\text{Max}\left|\prod_{i=1}^{N} H_i(z)\right| < 1 \qquad N=1,\ldots..,L-1 \qquad 3.4$$

The scaling property of Equation 3.4 generates maximum peaking of the magnitude response of the section with poles closest to the unit circle. The ordering rule, therefore, is to start with sections that are least peaked and move towards the most peaked.

The Matlab Signal Processing toolbox provides a function tf2sos (transfer function to second order section convert) that can be used to generate second order sections that are ordered according to the above rules based on appropriate optimum pole-zero pairing. This function generates a matrix sos containing the coefficients of each second order section of the equivalent transfer function H(z). The sos is a L x 6 matrix of the form

$$\mathbf{sos} = \begin{bmatrix} b_{01} & b_{11} & b_{21} & a_{01} & a_{11} & a_{21} \\ b_{02} & b_{12} & b_{22} & a_{02} & a_{12} & a_{22} \\ .... & & & & & \\ b_{0L} & b_{1L} & b_{2L} & a_{0L} & a_{1L} & a_{2L} \end{bmatrix} \qquad 3.5$$

The rows in Equation 3.5 are ordered such that the pole pair furthest from the unit circle and its nearest zero pair is in the first row of the matrix sos (i.e. k=1). A second-order cascade form structure of the IIR filter representing the SOS matrix of Equation 3.5 is shown in Figure 3.1. Note that the coefficient a_{0n}=1 for all the sections.

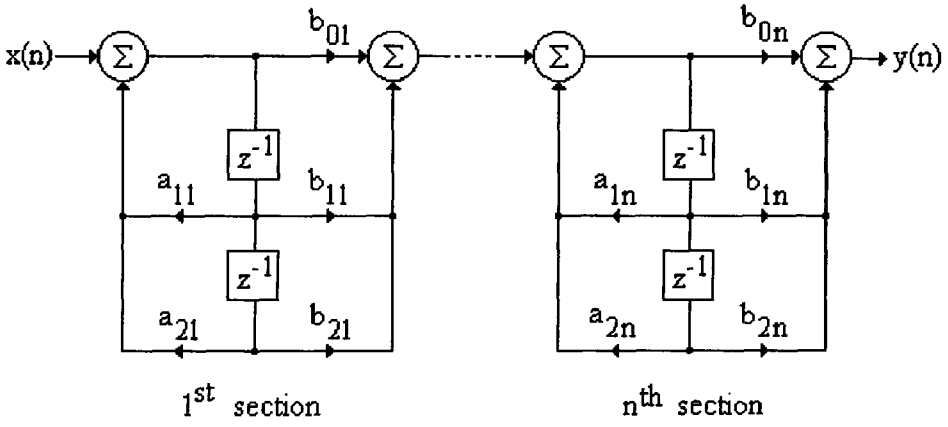

Figure 3.1 Cascade form structure of IIR digital filter

A FIR digital filter comprises of bounded impulse response coefficients and so is always stable. However, an IIR digital filter includes a feedback component that structurally cannot be guaranteed to be stable. Additionally, an IIR filter that is designed to be stable using infinite precision coefficients may become unstable when the coefficients are finite word length quantised. The testing of an IIR filter transfer function for stability is thus an intricate and integral part of the design procedure of such filters for real-time applications.

For high order transfer function the analytical calculation of pole positions is difficult thus the use of computer programs is essential for working out the roots of the denominator polynomial of the rational transfer function H(z) of Equation 3.2. Also, since 2nd order cascades of higher order IIR filters are less sensitive to quantised finite word length coefficients, the analysis for stability tests will consider a second-order transfer function.

The denominator of the transfer function H(z) assuming $a_0 = 1$, is given by

$$D(z) = 1 + a_1 z^{-1} + a_2 z^{-2} \qquad 3.6$$

The shaded region of the stability triangle shown in Figure 3.2 gives the region where the two conditions of coefficients a_1 and a_2 are satisfied i.e. $|a_2| < 1$ and $|a_1| < 1 + a_2$.

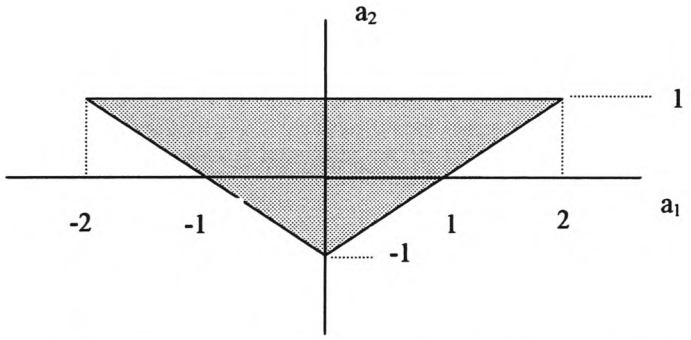

Figure 3.2 Stability triangle for a second-order transfer function.

Stability tests can be conducted in Matlab using the 'roots' function to obtain the poles of the denominator polynomial of the rational transfer function H(z). For GA optimisation, the individuals failing the stability test are awarded a large penalty for their objective function thereby leading towards stable solutions of the IIR filters.

3.3 GA optimisation of IIR digital filters

The design criteria of IIR digital filters are normally based on the assumption that the coefficients are of infinite precision and thus the system is a linear system. However, due to finite word-length constraint for fixed-point implementation of recursive filters, the digital filter becomes a non-linear system. This non-linearity causes a problem for developing analytical methods for optimisation of quantised coefficient IIR digital filters. No such method for optimisation has been reported in literature. However, computer programs have been developed for determining suitable minimum coefficient word-lengths for satisfying specific frequency response constraints [Ifeachor and Jervis, 1993 pp. 429].

Genetic algorithms are a useful tool in the optimisation of finite word-length constrained coefficients for realisation of IIR digital filters. Some examples of GA optimisation applied to the IIR filter design problems have been reported in literature. Wilson and Macleod [1993] consider a cascade form design of IIR filters. A simple GA is applied in order to find a compromise solution to the frequency response error and adder cost. In addition, stability and minimum phase is guaranteed by analysing the genes and identifying positions of poles and zeros. If the root is outside the unit circle then this is moved by multiplication with an all pass filter. Quantising the coefficients then follows this procedure. This step is a restriction to direct optimisation of the realisable IIR filter. Harris and Ifeachor [1995] have considered an automated design procedure for IIR filters. This work considers a hybrid GA approach that optimises second order cascade sections of IIR filters in terms of pole-zero positions on the z-plane. The GA culminates when an appropriate filter is located within specified bounds of maximum passband ripple and minimum stop band attenuation. The stability criterion of the stability triangle is used to return unstable solutions with low fitness function. A multiple objective fitness function includes a weighted component of round-off noise due to the ordering

of the second order sections and the frequency response of the filter. This combination allows a compromise solution to be found based on the two variables that can be controlled by the design specifications. Arslan and Horrocks [1995] have reported other work in this field. Again, second order sections are considered here, using real-valued coefficients, that are arranged in a concatenated form represented by a string of cascade stages. The frequency response template is specified within minimum and maximum limits and the overall fitness is evaluated as a function of the deviation from the exact design frequency response of the filter. Stability checks are also conducted based on the stability triangle.

The above methods for GA optimisation of IIR filters generate good results, however, the procedures used are generally, not flexible. For example, in the work of Wilson et al [1993], the GA optimisation of a second-order cascade form IIR filter is based on a compromise solution of frequency response error and adder cost for digital filters implemented on DSP devices that do not include a dedicated multiplier. The tests for stability and minimum phase are conducted by analysing the genes and identifying the positions of poles and zeros. If any root falls outside the unit circle thus contradicting the stability or minimum phase constraints, then the root is moved by multiplication with an all-pass filter. The finite word length quantisation of coefficients then follows this procedure. This step is a restriction to the final optimised filter. The work reported by Arslan et al [1995] overcomes some of these limitations. A direct design of FWL quantised coefficients is considered although there is no explicit mention of the manner in which the initial design of the IIR filters is conducted. Also, there is no mention of scaling and ordering issues for second-order cascade structures. Furthermore, the culmination of the GA is based on conformation of the optimal response to the specified bounds of maximum pass band ripple and minimum stop band attenuation. For these reasons, no extensive quantifiable measures of performance of the optimisation process have been reported for a range of filter types. Some of these issues have been investigated in this study.

The main advantage of the work covered in this study is that the complete process for design and optimisation is Matlab based and can be executed sequentially by selecting the appropriate choice of the filter design and the optimisation parameters. This option offers flexibility of design using the standard Matlab Signal Processing toolbox functions that include issues about scaling, pole-zero pairing and ordering of the second-order sections. The GA used in this work is a Matlab Toolbox developed by Chipperfield et al [1993]. This toolbox was originally developed for Control Systems applications and has been adapted for IIR digital filter optimisation. The main GA functions such as ranking, crossover and reinsert were used without any change and new m-file function for calculating the fitness function was written.

Such a Matlab based integral approach to the initial design of the IIR filter and subsequent GA optimisation makes this procedure flexible to obtaining quantifiable metrics for a number of design specifications. For example, the Matlab function tf2sos converts the high order rational transfer function H(z) into its second order sections in a cascade. The pole-zero ordering in the default option 'UP' is such that the first row of the matrix sos of Equation 3.5 will contain poles closest to the origin and the last row will contain poles closest to the unit circle. This option allows for minimum coefficient sensitivity due to quantisation. The 'SCALE' option of the tf2sos function specifies the desired scaling of the gain and the numerator coefficients of all the second order sections in the cascade. The 'SCALE' options available are the L_2-norm, L_∞-norm and no scaling. A combination of the default 'UP' ordering of pole-zero pairs and L_∞-norm scaling minimises the probability of overflow error in the realisation of the IIR filter.

Matlab programs `iir_ga.m` and `sos_ga.m` execute the GA optimisation of finite word length coefficient, IIR digital filters of the direct form and of a cascade of secod order sections respectively. The m-file codes of these programs are shown in Appendices C1.1 and C2.1 respectively. The complete process of optimisation is achieved in a single stage implementation

using a single frequency response template of the exact magnitude response. The magnitude error function is calculated using the following

$$\text{mag\_error} = \sum_{i=1}^{L} |H_{ei} - H_i| \qquad 3.7$$

where

H_{ei} = Magnitude response of the exact (i.e. high precision coefficient) filter at frequency 'i'

H_i = Magnitude response of the test filter at frequency 'i'

The phase error function is similarly calculated using the phase response of the filter. The object function value that must be minimised is evaluated using the following

$$\text{Obj\_Val} = \text{mag\_error} + W \ \text{phase\_error} \qquad 3.8$$

where W = weighting fraction

The percentage error between the exact and the best optimised magnitude response is calculated using the following

$$\%\ \text{mag\_error} = \frac{\sum_{i=1}^{L} |H_{ei} - H_i|}{\sum_{i=1}^{L} |H_{ei}|} 100 \qquad 3.9$$

3.3.1 The methodology and pseudo GA code for IIR filters

The simple genetic algorithm used in the optimisation of FWL quantised coefficients of IIR filters is based on the standard techniques of generating the initial population of individuals followed by objective function calculation, ranking and crossover. No mutation operator was included in the algorithm, as initial tests indicated no beneficial outcome of this operator for this optimisation problem. The GA used is a Matlab based toolbox designed and developed by

Chipperfield et al [1993]. A number of standard functions are included in the toolbox and are indicated by bold letters here. The description of each stage of the GA process is as follows.

1) Design of IIR filters

The design option selected in this application is the 'ellip' function of the Signal Processing toolbox of Matlab. This function is based on the Elliptical or Cauer IIR filter design algorithm. Such filters offer steeper roll-off characteristics than the Butterworth or the Chebyshev filters but are equiripple in both the pass and stop bands. In general, the elliptical filters meet the given specifications with the lowest order when compared with other filter types. The Matlab function for the elliptical IIR filter design uses the format

$$[b,a]=ellip(n, R_p, R_s, W_n, 'ftype') \qquad 3.10$$

Where 'b' and 'a' are the derived numerator and denominator coefficients of the IIR filter, 'n' is the order of the filter, R_p and R_s are the pass-band and stop-band allowable ripples (in dBs) respectively and W_n represents the cut-off frequency for low and high pass filters. When W_n = $[w_1 \ w_2]$, then 'ellip' returns an order 2n band pass filter with pass band $w_1 < \omega < w_2$. Note that W_n is a number between 0 and 1 where 1 corresponds to the Nyquist frequency. 'ftype'=high defines a high pass filter with cut-off frequency W_n.

The study in this work covered the FWL coefficient GA optimisation of the direct form and the second-order cascade form of IIR filters. The coefficients 'b' and 'a' derived from Equation 3.10 generate the direct form version of the IIR filters. For representation in the second-order cascade form, the transfer function must be converted appropriately by using the Matlab function of the form

$$sos = tf2sos(b, a, 'order', 'scale') \qquad 3.11$$

Where 'order' specifies the ordering of the second-order sections of Equation 3.5. If order = 'up', then the first row of Equation 3.5 will contain the poles closest to the origin and the last

row will contain poles closest to the unit circle. If 'order' = 'down', then the sections are ordered in the opposite direction. The zeros are always paired with the poles closest to them. The 'scale' option specifies the desired scaling of the gain and the numerator coefficients of all the second-order sections. The options available in the Matlab function are; 'none', 'two' and 'inf' representing 'no scaling', L_2 norm and the L_∞ norm scaling respectively. These various options for ordering and scaling offers flexibility of design and preferences to the design engineer for an appropriate choice based on the specific application. However, the following two options are likely to be most useful for the design of second-order cascade form IIR filters [Jackson, 1989].

- Using the infinity norm scaling in conjunction with the 'up' ordering will minimise the probability of overflow in the realisation.
- Using the 2-norm scaling in conjunction with the 'down' ordering will minimise the peak round-off noise.

The option of infinity-norm scaling in conjunction with the 'up' ordering generally offers the best compromise and it also allows the effects of scaling to be verified experimentally using sinusoidal input signals [Ifeachor and Jervis, 1993]. For this reason, the FWL coefficient quantisation considered for optimisation in this study is based on this option of infinity norm scaling and 'up' ordering of the second-order sections. For completeness, the results of another option of 'no-scaling' and 'up' ordering is also included in the study here. It must be recognised that there is no loss of generality of the GA optimisation process if the initial design of the FIR filter is based on other standard design techniques such as the Butterworth, Chebyshev or the Yulewalk functions available in the Matlab Signal Processing toolbox.

2) Representing real-valued coefficients with FWL rounded values

The design process for the IIR filters generates a set of real-valued coefficients of the form shown in Equation 3.1. The next step is to obtain the rounded integer valued coefficients from the real-valued coefficients. As discussed for the case of FIR filters in Chapter 2, the IIR filters considered here are specifically designed for their realisation on fixed-point devices that have the advantage of efficient computational throughput, low cost and low power dissipation when compared with their floating point counterparts. The study in this work is thus restricted to the optimisation of quantised FWL coefficients for fixed-point devices and the calculations involved are based on the fixed-point arithmetic. The method used for deriving the FWL coefficients of IIR filter for fixed-point arithmetic is identical to that used for the case of FIR filters covered in Chapter 2. For convenience, this is repeated here. The high-precision coefficients derived through an appropriate design stage for the IIR filters are in the form of real valued numbers. The most commonly used method of deriving FWL coefficients for fixed-point arithmetic is the direct quantisation method. In this method, the high precision coefficients that are derived using standard filter design techniques are first rounded to yield FWL quantised coefficients. The starting solution of quantised coefficients is thus given by

$$h_{ri} = \text{round}[h_{ei}\, 2^{B-1}] \qquad 3.11$$

Where 'h_{ri}' is the rounded coefficient, 'h_{ei}' is the high precision coefficient and 'B' is the number of bits used to represent the coefficients.

3) Generating initial population

The first step towards generating a population set of individuals is to start with the rounded coefficient values. The initial string is thus obtained by concatenating the rounded integer-valued coefficients 'b' and 'a' in the form x = [b, a]. The population set of individuals is then

obtained by randomly perturbing each rounded integer-valued coefficient by +1, 0 or −1. This range of perturbation is obtained using the base value BASE=1. The choice for a variation of +1, 0 or −1 is simply to do with the word length of the quantised coefficients. For example if we consider a word length of 6 bits, then the real valued coefficients can take integer values ranging from +31 to −32 in a two's complement format where the most significant bit is a sign bit. A change of +1 then amounts to a fractional change of $1/2^5 = +0.03125$ of the real valued coefficient. Increasing the base value to say BASE=2 can extend this range and thus the search space. The random perturbation of coefficients will then be +2, +1, 0, -1 or −2. It must be mentioned that an appropriate choice of the base value depending on the filter length and the number of bits being used to represent the coefficients, is an area that is not fully investigated. For this study, an initial trial of several different filters using base value of 1, 2 and 3 was conducted. The test results for a population size of 100 over 20 generations, consistently generated good results for base value of 1. An extensive range of search space could have been tested over larger population size and greater number of generations. However, the motivation for using GAs in this study was to test this optimisation process as a general framework against other methods and to draw a comparative measure.

4) Objective function evaluation

The main purpose of the optimisation process is to minimise the objective function with the specific aim of obtaining an approximated frequency response of the filter that is as close as possible to the desired response. The discrete search space for the example filters considered in this Chapter could be calculated using the filter lengths and the base value used for coefficient perturbations. The filter length ranges from 10 to 18 for the direct form IIR filters and from 12 to 24 for the second-order cascade form IIR structures. The discrete search space for a base value of 1 is then 3^{10} for filter length 10 and 3^{18} for filter length 18 for direct form IIR filters and 3^{12} and 3^{24} for filter lengths 12 and 24 respectively of the second-order cascade form structures.

This search space increases substantially when the base value is increased to 2. The GA used for optimisation conducts 120 objective function evaluations initially followed by 100 evaluations over 20 generations. This makes a total of 2120 objective function evaluations.

The objective function is calculated for L=500 equally spaced frequency grid points. An example low pass filter specification of normalised frequency scale where Nyquist frequency = 1.0, is of the form

Pass band: maximum allowable ripple = 1dB

Stop band: stop band attenuation = 40dB

W_n (cut-off frequency): = 0.5 = half the Nyquist frequency.

The magnitude error function is calculated using the following

$$\text{mag\_error} = \sum_{i=1}^{L} |H_{ei} - H_i| \qquad 3.12$$

where

H_{ei} = Magnitude response of the exact filter at frequency 'i'

H_i = Magnitude response of the test filter at frequency 'i'

The phase error function is similarly calculated using the phase response of the filter. The object function value that must be minimised is evaluated using the following

$$\text{Obj\_Val} = \text{mag\_error} + W \ \text{phase\_error} \qquad 3.13$$

where W is the weighting factor.

After some initial trials it was observed that a value of W=0.001 generated good compromise solutions for the magnitude and phase response of the optimised filters.

It must be recognised that stability of the optimised design must be assured for a realisable IIR filter. The objective function sub-code that is called by the main GA code incorporates stability checks for every individual by deriving the roots. Individuals failing the stability test are awarded a high penalty of the objective function value and are thus effectively eliminated from the overall search.

5) Fitness value and ranking

The Matlab based **ranking** function of the GA toolbox ranks the individuals according to their objective function values 'ObjV' and returns a column vector consisting of the corresponding fitness value 'FitnV' of the individuals. This function performs a linear ranking with a selective pressure (SP) of 2 [Whitley, 1998]. The fitness value assigned to the individuals is calculated according the following formula given by Equation 1.1 in Chapter 1.

6) Selection of individuals for breeding

The high-level function for selection of individuals from the population set and returning the selected individuals in a new population is performed by the **select** function. The low-level selection function **sus** is called by the **select** function. The **sus** function is based on a form of stochastic sampling method and is implemented by obtaining a cumulative sum of the fitness vector 'FitnV' and generating a set of equally spaced numbers between 0 and Σ(FitnV) [Baker, 1987]. The probability of an individual being selected is given by Equation 1.2 in Chapter 1.

7) Recombining individuals – crossover

The crossover function is also performed in two stages. The high-level function is **recombin** that calls the low-level function **recdis**. The **recdis** function is a discrete recombination function. The mating process is performed between pairs of rows. The **recdis** function first generates an internal mask table that determines which parents contribute which variables to the

offspring. On the basis of the randomly generated mask table, the variable values are exchanged between the individuals and return a new population after mating.

8) Reinsert offspring into new population

The new population set generated after crossover is subjected to the objective function evaluation of each new individual. On the basis of their fitness, the offspring are selected for reinsertion into the new population. The objective function values are then copied according to the reinserted offspring and the GA loop is then repeated for the next generation.

A pseudo GA code for FIR filter optimisation and the objective function code are shown in Figures 3.3 and 3.4 respectively.

3.3.2 Example IIR filters

A number of example IIR filters were used to test the robustness of the GA optimisation code. The filters are tested for the direct form format and the second-order cascade form for a number of different types, order and number of bits. Table 3.1 shows the different types of filters used i.e. A-type is a low pass filter with cut-off frequency $W_n = 0.5\pi$, B-type is a band pass filter with cut-off frequencies $W_n = [0.3\pi \quad 0.7\pi]$ and C-type is high pass filter with cut-off frequency $W_n = 0.6\pi$. Each of these filters were tested for 4^{th}, 6^{th} and 8^{th} order using 5, 8 and 12 bits in each case. Further distinction of filters used were to test for minimal phase and non-minimal phase in the case of direct form filters and for the second order cascade form the case of 'Infinity norm' and 'No norm' options were both used. The GA specifications used are

Population size (Nind) =120
Maximum number of generations =20
Weighting fraction (W) = 0.001
Number of frequency points (L) = 500
Generation gap (GGAP) = 0.8
Insertion rate (INSR) = 1

```
%Pseudo GA code for FWL coefficient optimisation of IIR digital filter
% GA characteristics

% IIR Filter design procedure
% could use several filter design functions such as
% butterworth, cheby, ellip, yulewalk etc.

 [b,a] = ellip(n,Rp,Rs,Wn); % for direct form realisation

% alternative second-order section realisation
% sos=tf2sos(b,a,'up','none');% 'up' ordering and 'None' norm
% sos=tf2sos(b,a,'up','inf'); % 'up' ordering and 'Infinity' norm
% sosba=sos;

[h,w] = freqz(b,a,L);       % frequency response
h = abs(h);                 % magnitude response
p = angle(freqz(b,a,L));    % phase response
x = [b,a];

% filter with coefficients simply rounded

% create new population
Chrom = iir_pop(Nind,xr,xmask,BaseV,n); % create population
gen = 0                         % generational counter
% work out the object function value for each individual
ObjVal = iir_objf(Chrom,n1,h,hr,p,x,L);

% best chromosome for minimum object function value
[Best(gen+1),ix] = min(ObjVal);
xcbest = Chrom(:,ix);

% start of generational loop
while gen < MAXGEN
   % assign fitness value to each individual in population
   FitnV = ranking(ObjVal);
   % select good individuals for breeding
   SelCh = select('sus',Chrom,FitnV,GGAP);
   % recombine selected individuals - crossover
   SelCh = recombin('recdis', SelCh, 1);
   % evaluate object function of offsprings
   ObjVOff = feval('iir_objf',SelCh,n1,h,hr,p,x,L);
   % reinsert good offsprings into current population
   [Chrom, ObjVal] = reins(Chrom, SelCh, 1, [1 INSR], ObjVal,
ObjVOff);
   %increment generational counter
   gen=gen+1
   % update display and remember the best individual to date
   [Best(gen+1,1),ix] = min(ObjVal);
   xcbest = Chrom(:,ix);
end
% end of GA
```

Figure 3.3 Pseudo GA code for finite word length coefficient optimisation of IIR digital filter

```
% iir_objf.m
% Calculate object function value for GA code iir_ga.m
function ObjVal = iir_objf(Chrom,n1,h,hr,p,x,L);

% work out population parameters
[Nind,Nvar] = size(Chrom);

% start of loop for each individual
for irun = 1:Nvar;

% calculate coefficients ac and bc
 xc = Chrom(:,irun);
 bc = xc(1:n1);
 ac = xc(n1+1:2*n1);

% work out the roots of xc
g1 = [abs(roots(ac))];
g2 = [abs(roots(bc))];

% work out the absolute magnitude and angle
hc = abs(freqz(bc,ac,L));
pc = angle(freqz(bc,ac,L));

% stability criteria checked, if any root
% is > or = 1, then replace hc with large hc
if (any(g1 >= 1))
   hc = 100 + hc;
end

% minimum phase checked, if any root
% is > or = 1, then replace hc with large hc
if (any(g2>=1))
  hc = 100 + hc;
end
% work out magnitude and phase errors
mag_err = sum(abs(h-hc));
ph_err = sum(abs(p-pc));
m=floor(length(p)/2);

% work out a weighted object value between
% magnitude and phase errors

ObjVal(irun,:) = mag_err + 0.001*ph_err;

% optional alternative objective value
% ObjVal(irun,:) = mag_err;
end
```

Figure 3.4 Objective function called by the GA code for finite word length coefficient optimisation of IIR digital filter

Table 3.1

Set of IIR Filter Specifications

Note that the 'ellip' function of Matlab specifies cut-off frequencies W_n between 0 and 1 where 1 is the Nyquist frequency.

Filter	W_n
A: range (low pass)	0.5
Pass band ripple:	1 dB
Stop band attenuation:	40 dB
B: range (band pass)	[0.3 0.7]
Pass band ripple:	1 dB
Stop band attenuation:	40 dB
C: range (high pass)	0.6
Pass band ripple:	1 dB
Stop band attenuation:	40 dB

3.3.3 IIR filters – direct form

The results for direct form optimisation of IIR filters for various types, order and number of bits used are shown in Table 3.2. The key to filter representation e.g. DF/HP8/12 means direct form, high pass filter, 8^{th} order using 12 bits to represent the coefficients. The GA optimised results MP represents minimum phase and NMP represents non-minimal phase. 'Rounded' results represent sum of magnitude error for simply rounded coefficient values. Both the GA optimised and the rounded results are obtained using the absolute value of the sum of error given by Equation 3.14 over the full frequency range with 500 points.

Table 3.2

GA optimisation results of summed magnitude error of Equation 3.14 over 500 frequency points for direct form IIR filters. MP = minimal phase and NMP = non minimal phase

Filter	Rounded	GA-op MP	GA-op NMP
Low-pass			
DF/LP4/5	56.8062	25.2991	16.5262
DF/LP4/8	2.6726	1.0608	1.0608
DF/LP4/12	0.1300	0.1300	0.1300
DF/LP6/5	103.4716	72.7669	45.5413
DF/LP6/8	18.8755	17.7590	6.0774
DF/LP6/12	1.2700	2.2991	0.7048
DF/LP8/5	654.8179	654.8179	654.8179
DF/LP8/8	1.72e+14	32.0719	27.1850
DF/LP8/12	5.8895	6.4482	5.8895
High-pass			
DF/HP4/5	71.4119	35.3775	20.1340
DF/HP4/8	5.9536	4.0801	3.0417
DF/HP4/12	0.3098	0.2450	0.1464
DF/HP6/5	193.5966	193.5966	155.8164
DF/HP6/8	44.5597	19.2983	19.2983
DF/HP6/12	7.3556	5.7735	1.4430
DF/HP8/5	None	None	None
DF/HP8/8	None	None	None
DF/HP8/12	33.4680	23.7925	14.6120
Band-pass			
DF/BP4/5	23.3878	13.3873	12.2060
DF/BP4/8	2.0337	1.3471	1.3471
DF/BP4/12	0.1042	0.0567	0.0567
DF/BP6/5	28.3810	14.1170	14.1170
DF/BP6/8	4.4943	2.9954	0.6135
DF/BP6/12	0.1709	0.0882	0.0821
DF/BP8/5	71.4771	35.4355	25.7093
DF/BP8/8	5.9575	5.9575	3.0478
DF/BP8/12	0.3112	0.3112	0.2360

The GA optimisation for each run over 20 generations using 120 individuals takes approximately 2 minutes using a Pentium–III computer. Typical magnitude and phase response results obtained for various direct form IIR filters are shown in Figures 3.2 to 3.6. Figure 3.7 shows a typical result for a range of percentage error values obtained using Equation 3.16 against number of bits for a 6th order direct form low pass non-minimal phase filter. The GA code for direct form optimisation is shown in Appendix C1.1. A complete listing of coefficient values for exact filter, rounded coefficient filter, GA optimised filter with and without minimal phase constraints are shown in Appendix C1.2.

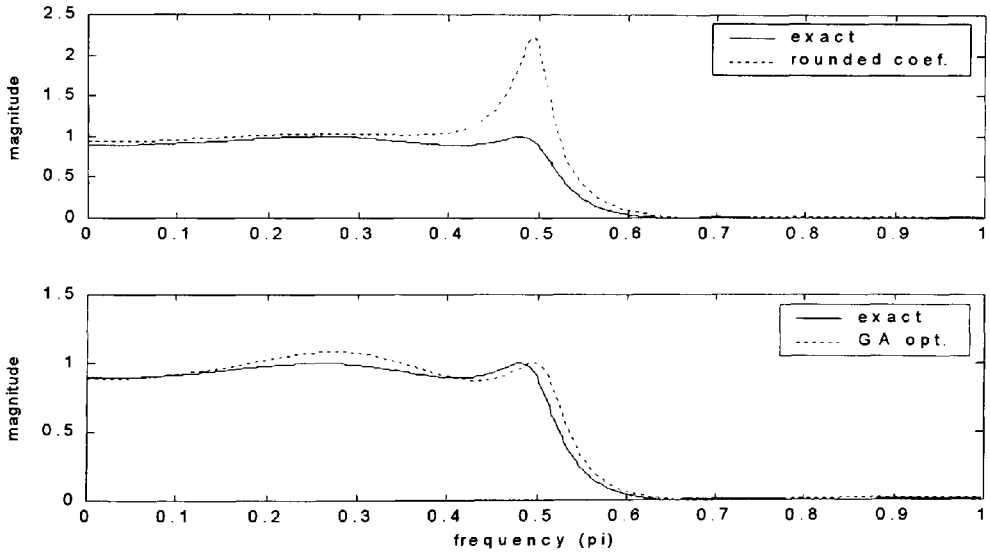

Figure 3.5 Magnitude response of a 4th order direct form low-pass filter using 5 bit coefficients and non-minimal phase.

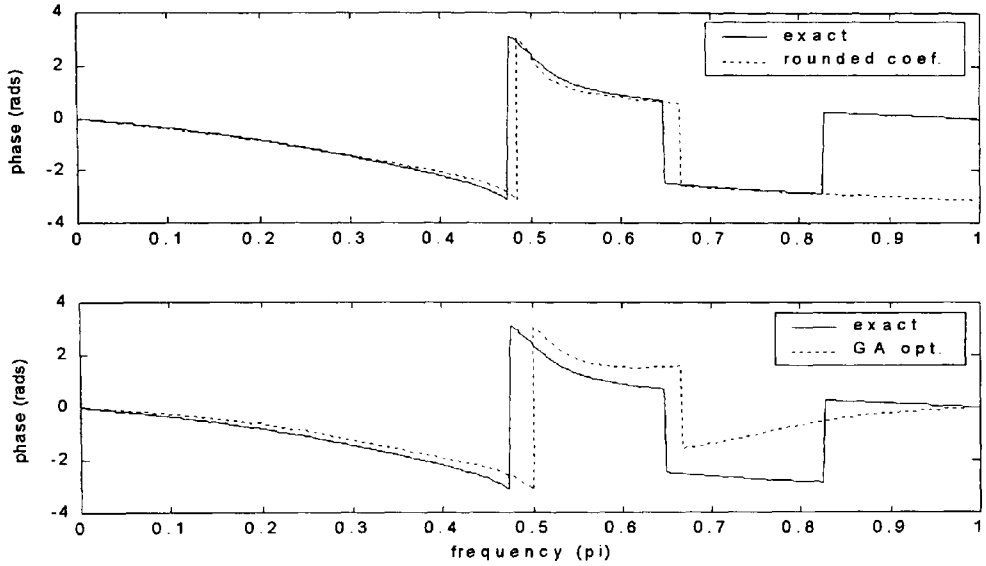

Figure 3.6 Phase response of a 4th order direct form low-pass filter using 5 bit coefficients and non-minimal phase.

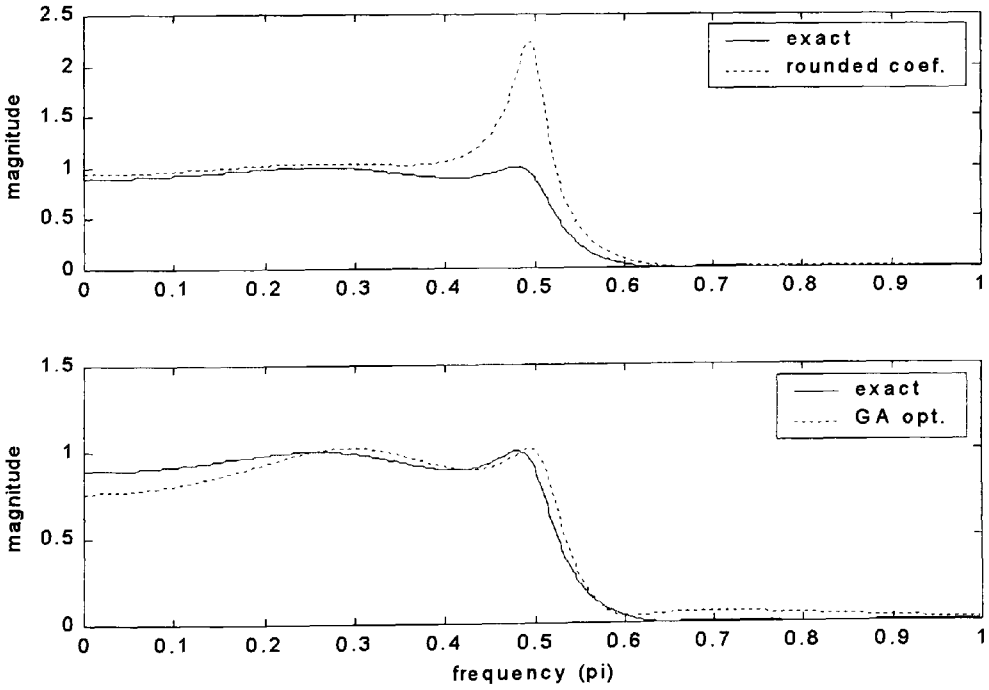

Figure 3.7 Magnitude response of a 4th order direct form low-pass filter using 5 bit coefficients and minimal phase.

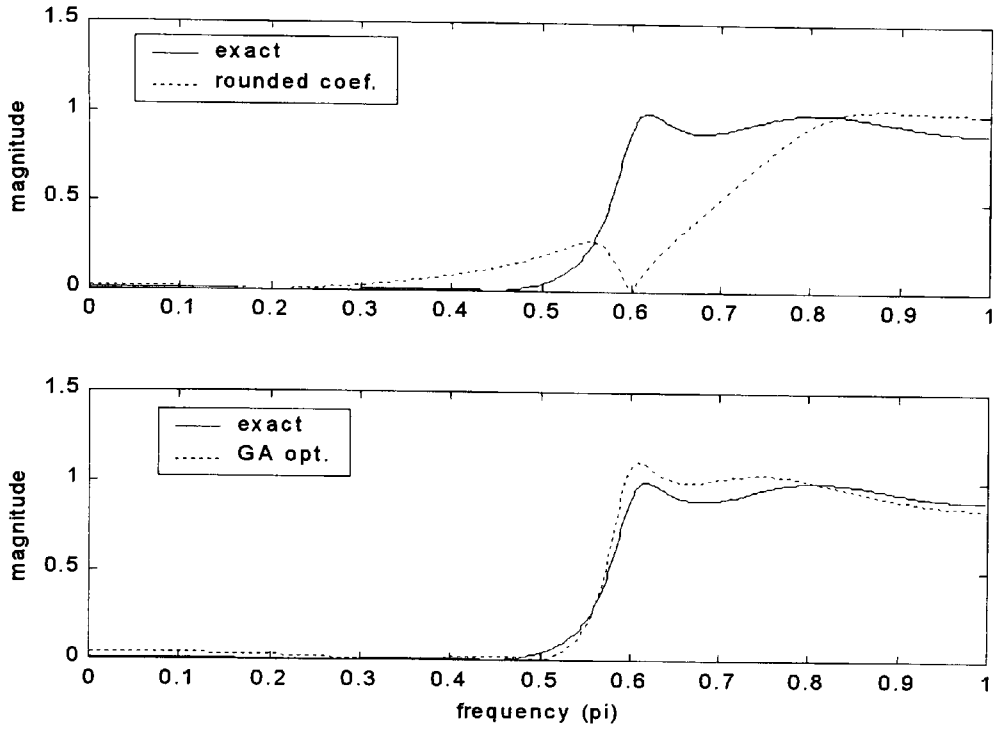

Figure 3.8 Magnitude response of a 4th order direct form high-pass filter using 5 bit coefficients and non-minimal phase.

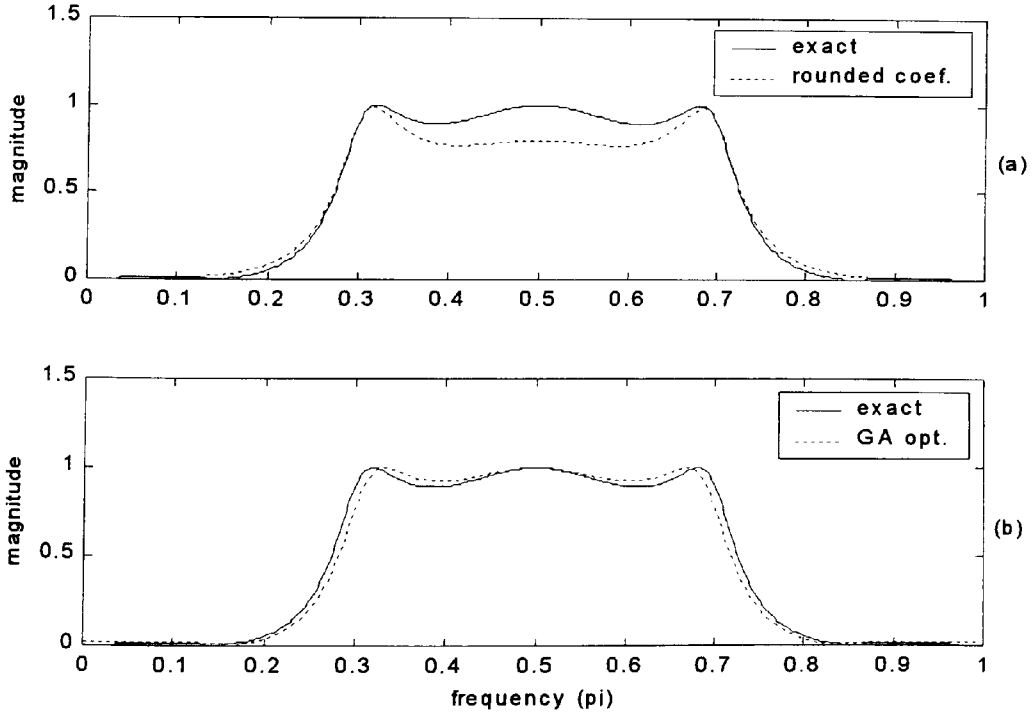

Figure 3.9 Magnitude response of a 6th order direct form band-pass filter using 5 bit coefficients and minimal phase.

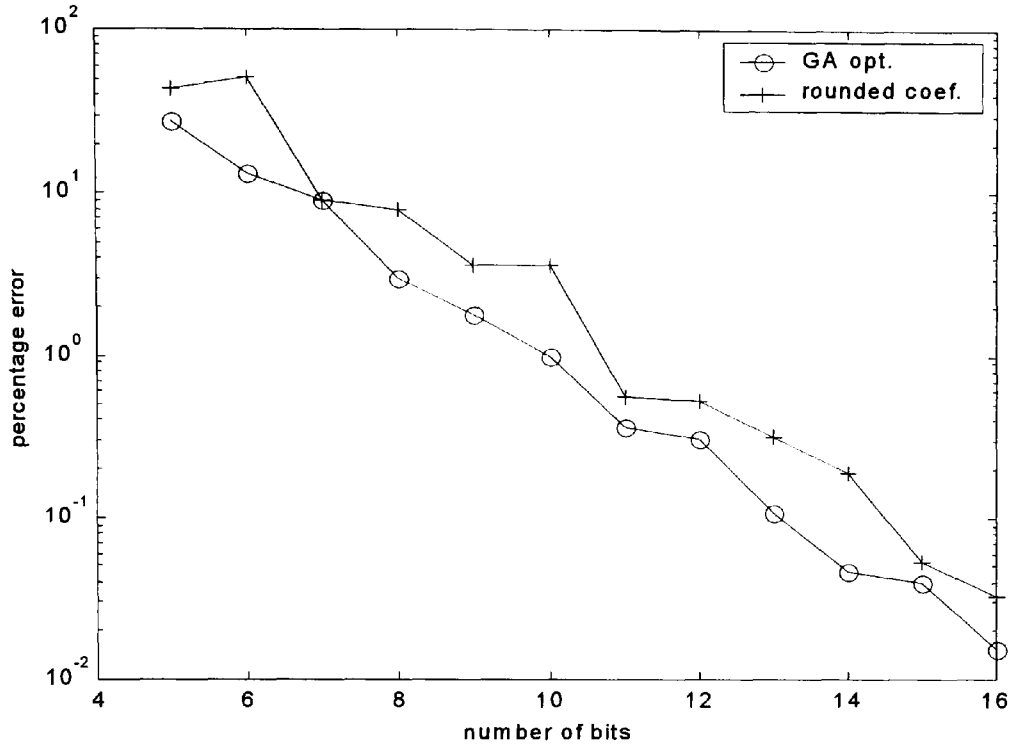

Figure 3.10 Magnitude response percentage error against number of bits of a 6th order direct form low-pass filter and non-minimal phase.

3.3.4 IIR filters – second order section form (SOS)

The results for second-order section form optimisation of IIR filters for various types, order and number of bits used are shown in Table 3.3. The key to filter representation e.g. SOS/BP6/8 means second-order section form, band pass filter, 6th order, using 8 bits to represent the coefficients. The GA optimised results NN represents 'None norm' and IN represents 'Infinity norm'. 'Rounded' results represent sum of magnitude error for simply rounded coefficient values both for the 'None norm' NN and the 'Infinity norm' IN as shown. Both the GA optimised and the rounded results are obtained using the absolute value of the sum of error given by Equation 3.14 over the full frequency range with 500 points. The GA optimisation for each run over 20 generations using 120 individuals takes approximately 2 minutes using a Pentium–III computer. Typical magnitude and phase response results obtained for some second

order cascade form IIR filters are shown in Figures 3.8 to 3.15. Figures 3.16 and 3.17 show results for a range of percentage error values obtained using Equation 3.16 against number of bits for the SOS/HP8 non-Norm and the SOS/HP8 Infinity norm filters respectively. The GA code for the second order cascade form optimisation is shown in Appendix C2.1. A complete listing of coefficient values for exact filter, rounded coefficient filter, GA optimised filter with 'Infinity norm' and with 'None norm' are shown in Appendix C2.2.

Table 3.3

GA optimisation results of summed magnitude error of Equation 3.7 over 500 frequency points for second order cascade form IIR filters. NN: None Norm, IN: Infinity norm

Filter	Rounded-NN	GA-op NN	Rounded-IN	GA-op IN
Low-pass				
SOS/LP4/5	42.6433	6.1447	78.7948	6.1447
SOS/LP4/8	2.2285	0.9172	4.9949	1.0973
SOS/LP4/12	0.2156	0.0644	0.1124	0.0467
SOS/LP6/5	13.7039	8.1174	90.2727	28.3454
SOS/LP6/8	3.3786	0.8437	9.9007	3.7464
SOS/LP6/12	0.1844	0.0535	0.6900	0.2449
SOS/LP8/5	4.3054e+012	10.3179	None	None
SOS/LP8/8	1.4933	1.1381	73.5649	18.9085
SOS/LP8/12	0.3155	0.0628	9.3633	1.9365
High-pass				
SOS/HP4/5	26.9774	10.3195	26.9774	10.3195
SOS/HP4/8	3.9268	1.4458	2.6164	1.3009
SOS/HP4/12	0.2118	0.0837	0.5764	0.0561
SOS/HP6/5	66.3240	12.0048	None	None
SOS/HP6/8	3.4268	1.5285	5.7506	4.0782
SOS/HP6/12	0.6927	0.0687	4.2936	0.3428

Table 3.3 continued

SOS/HP8/5	102.2992	12.8340	None	None
SOS/HP8/8	6.1425	1.6328	152.3241	21.4905
SOS/HP8/12	0.8247	0.0898	6.6315	1.8065
Band-pass				
SOS/BP4/5	114.8468	14.9344	115.2148	29.3585
SOS/BP4/8	5.8294	2.5332	11.6847	3.1483
SOS/BP4/12	0.4959	0.2074	0.4032	0.2462
SOS/BP6/5	114.4422	21.5173	191.9780	22.4432
SOS/BP6/8	4.6612	2.0228	25.9615	3.2291
SOS/BP6/12	0.7241	0.1354	0.8949	0.1546
SOS/BP8/5	110.0014	19.7059	152.6596	23.1082
SOS/BP8/8	15.5158	2.9377	32.5462	6.4900
SOS/BP8/12	0.4568	0.2097	0.8905	0.3449

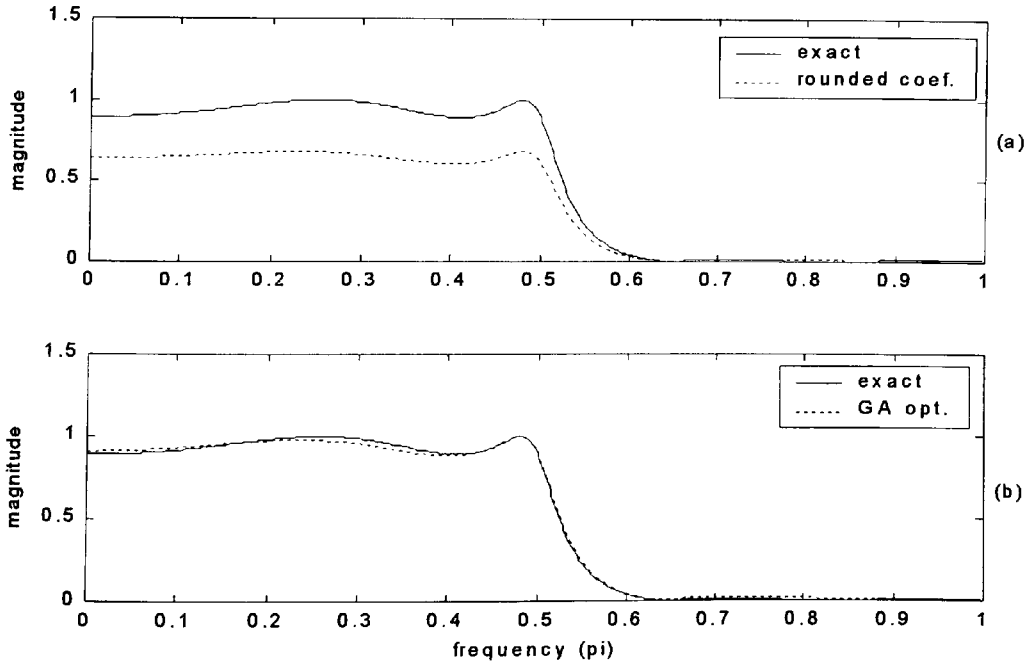

Figure 3.11 Magnitude response of a second order cascade form 4th order low-pass filter using 5 bit coefficients and Infinity norm..

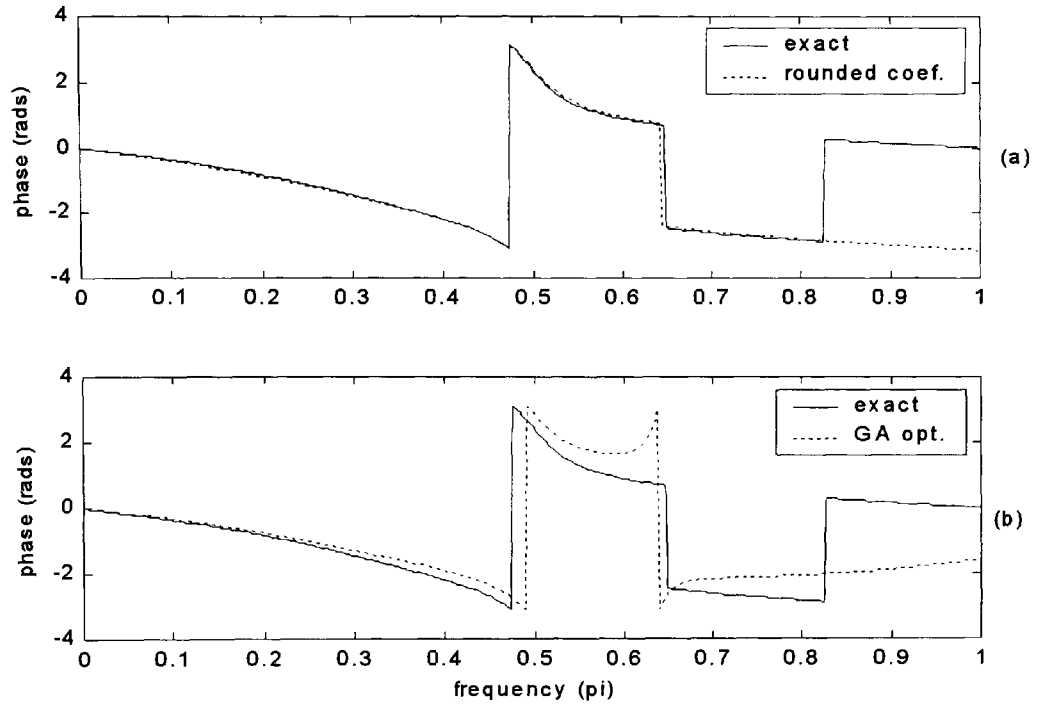

Figure 3.12 Phase response of a second order cascade form 4th order low-pass filter using 5 bit coefficients and Infinity norm.

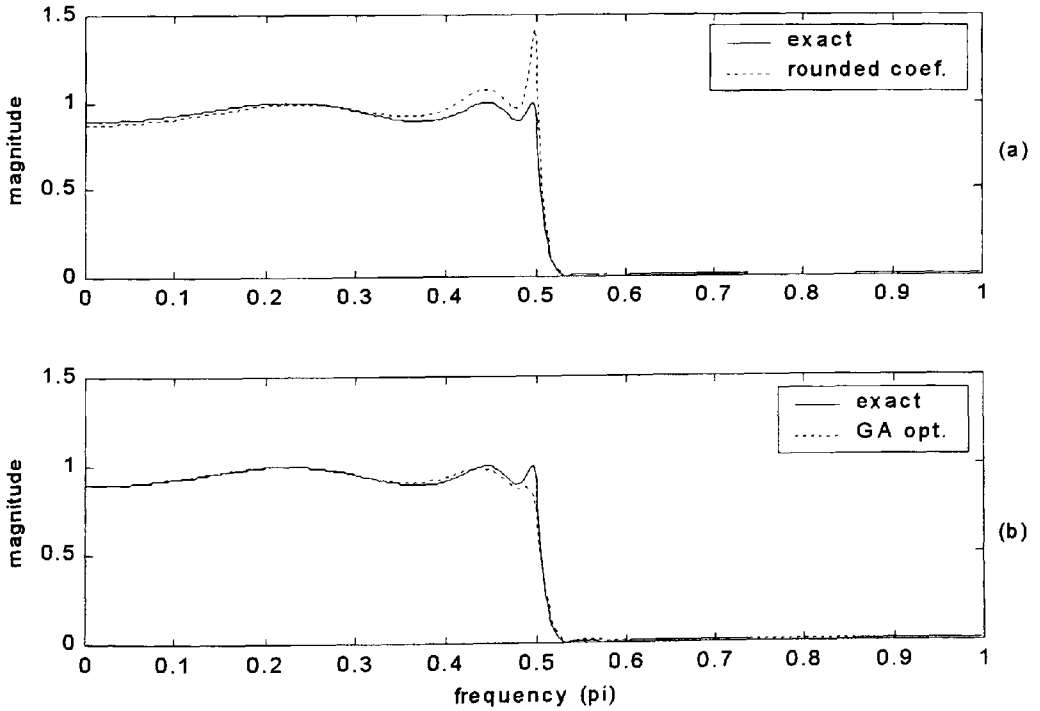

Figure 3.13 Magnitude response of a second order cascade form 6th order low-pass filter using 8 bit coefficients and Infinity norm.

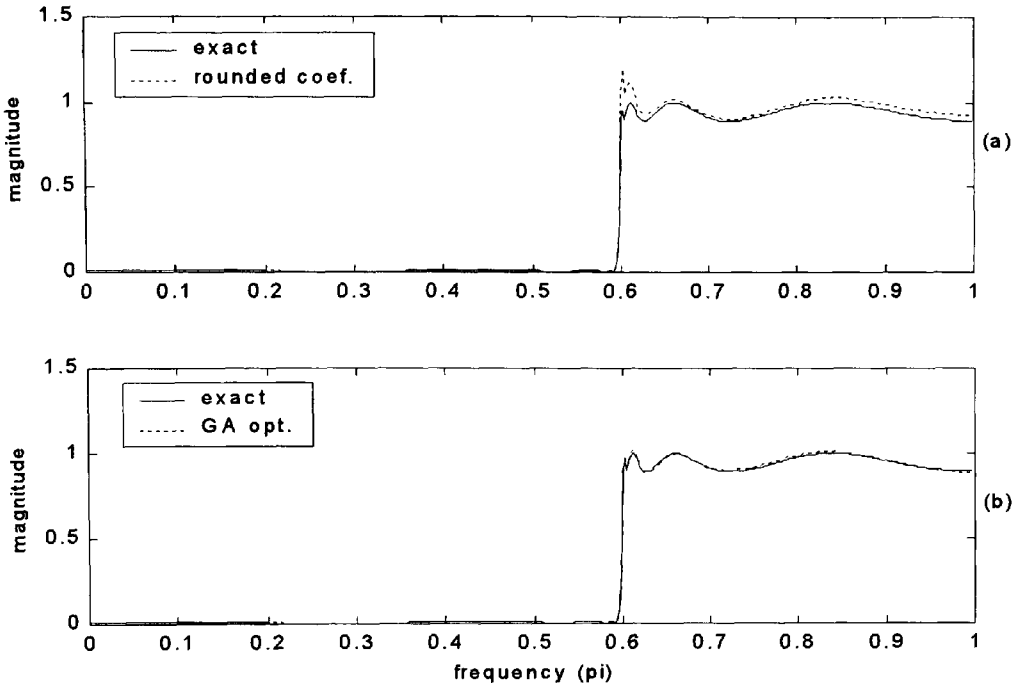

Figure 3.14 Magnitude response of a second order cascade form 8th order high-pass filter using 8 bit coefficients and 'None' norm.

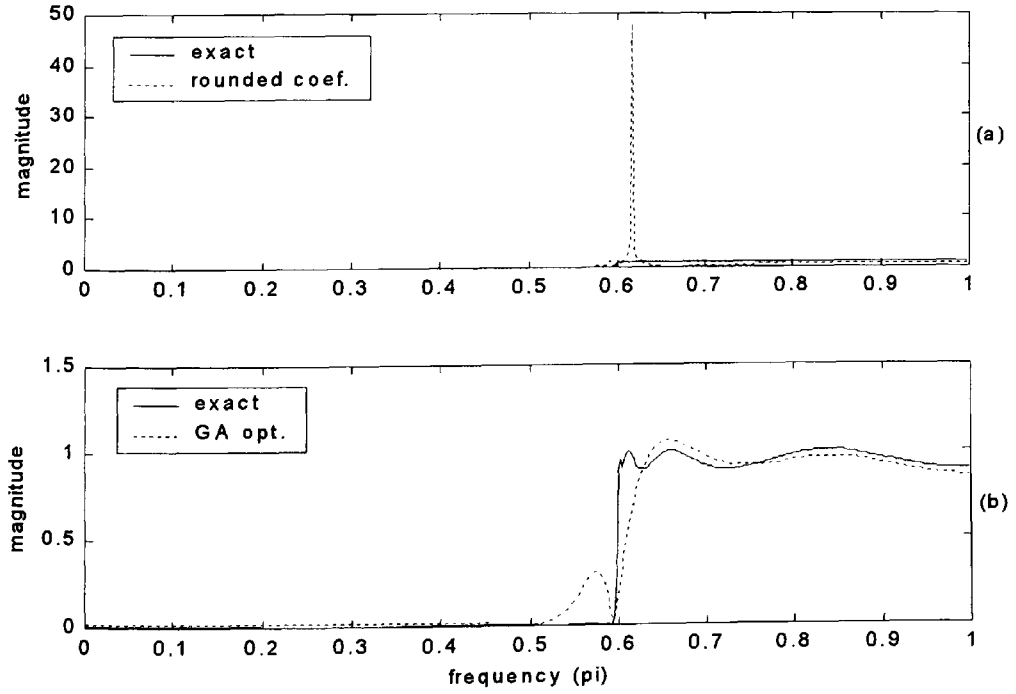

Figure 3.15 Magnitude response of a second order cascade form 8th order high-pass filter using 8 bit coefficients and 'Infinity' norm.

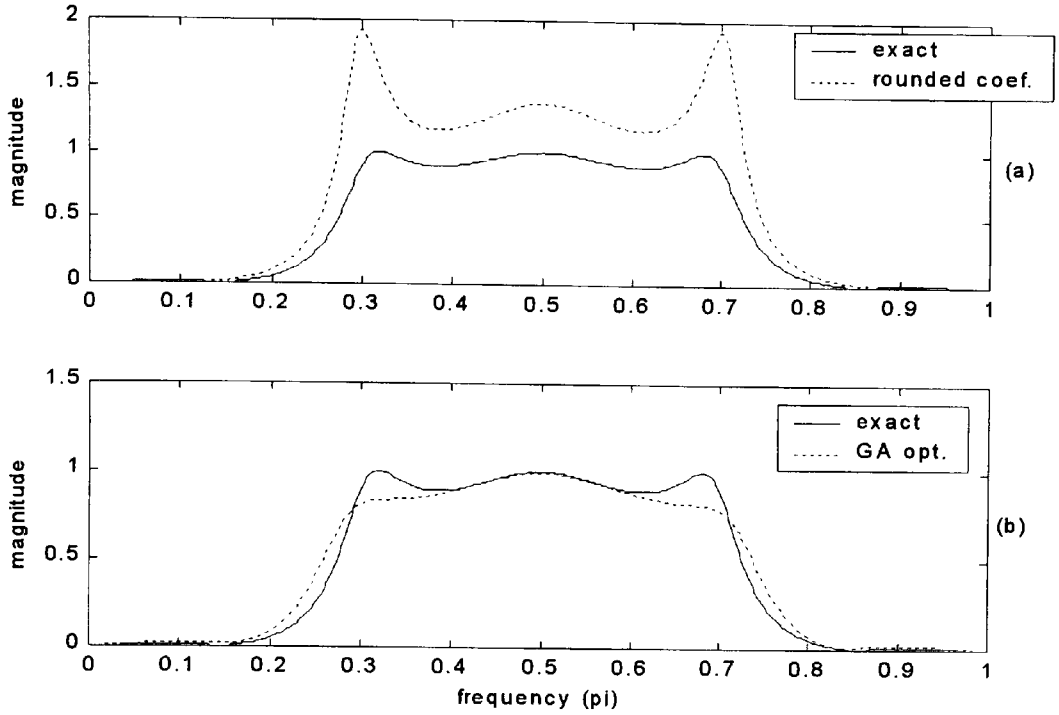

Figure 3.16 Magnitude response of a second order cascade form 6th order band-pass filter using 5 bit coefficients and 'none' norm.

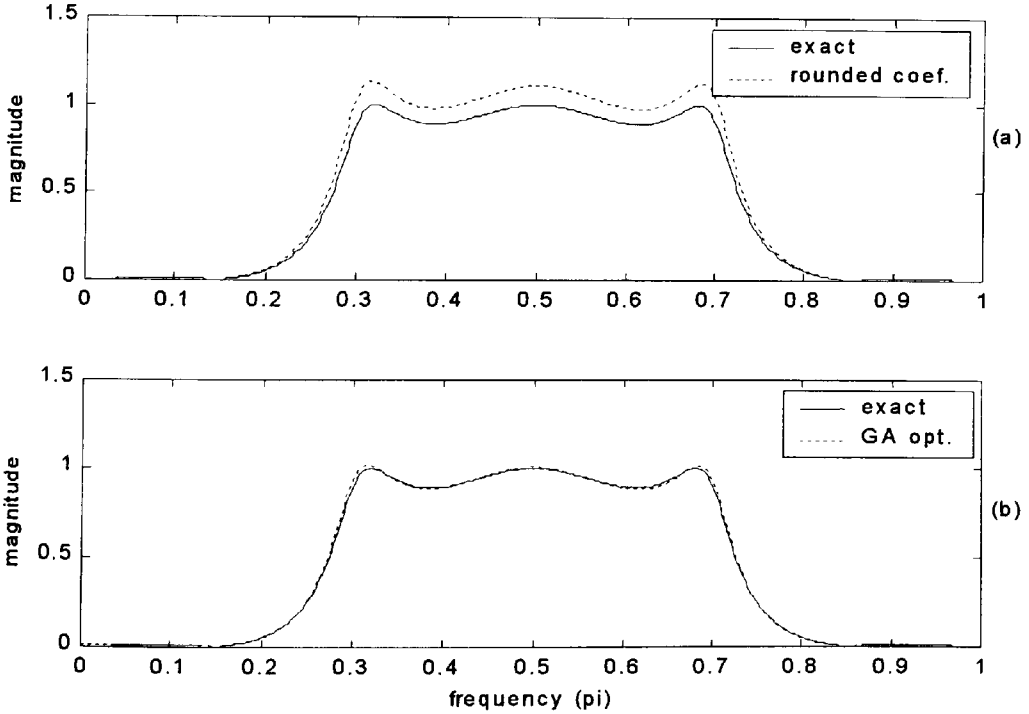

Figure 3.17 Magnitude response of a second order cascade form 6th order band-pass filter using 8 bit coefficients and Infinity norm.

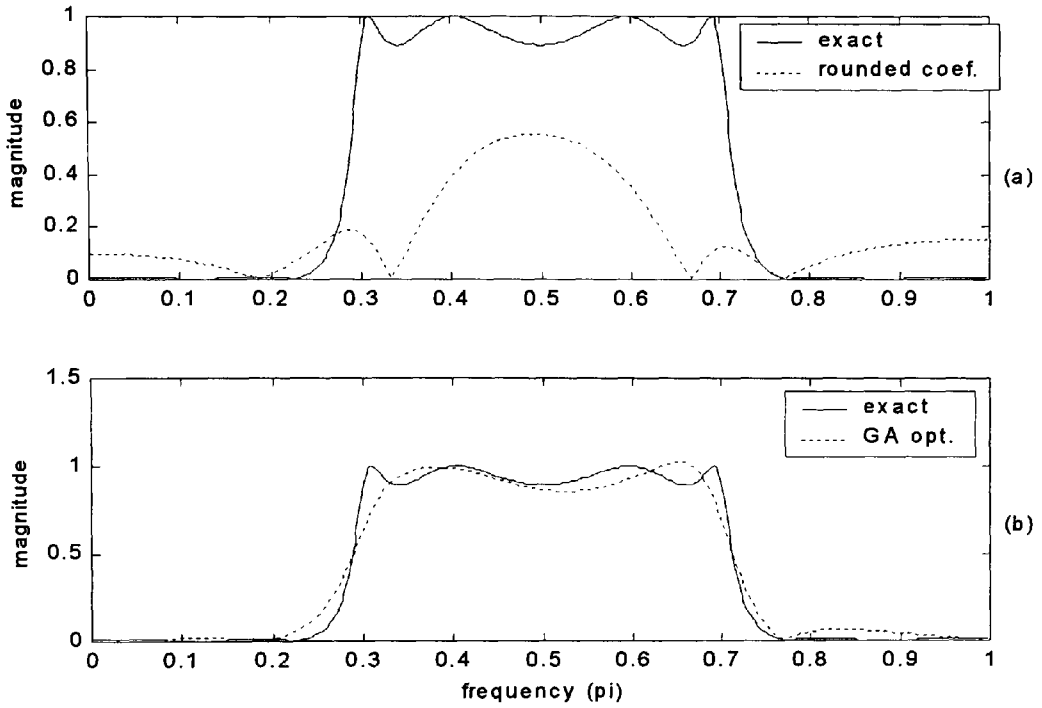

Figure 3.18 Magnitude response of a second order cascade form 8th order band-pass filter using 5 bit coefficients and Infinity norm.

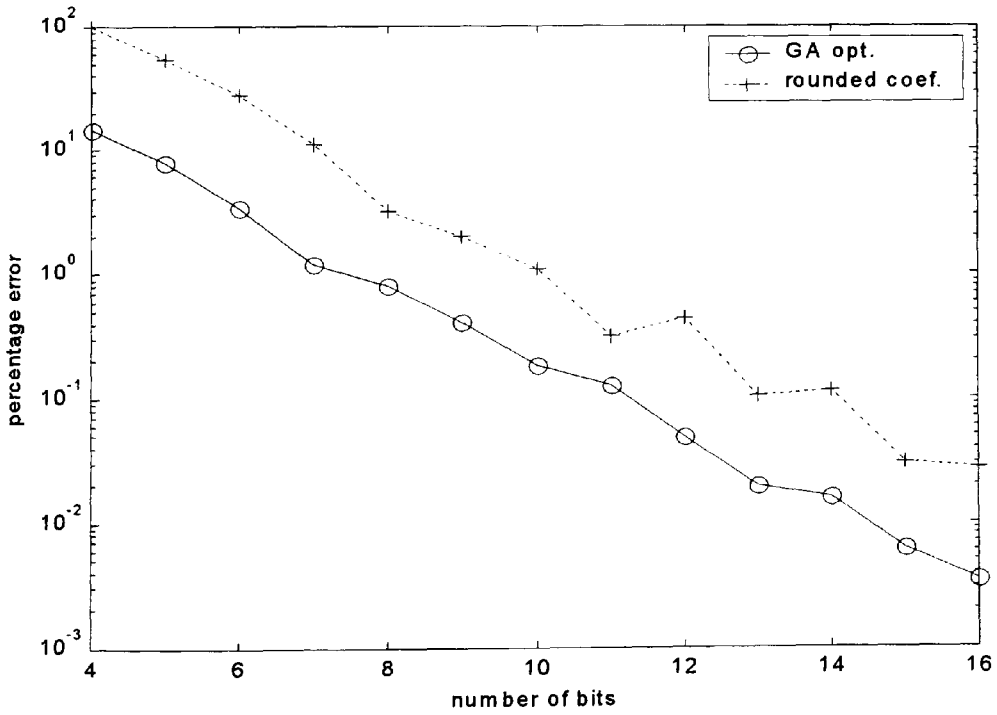

Figure 3.19 Magnitude response percentage error against number of bits of a second order cascade form 8th order high-pass filter with 'None' norm.

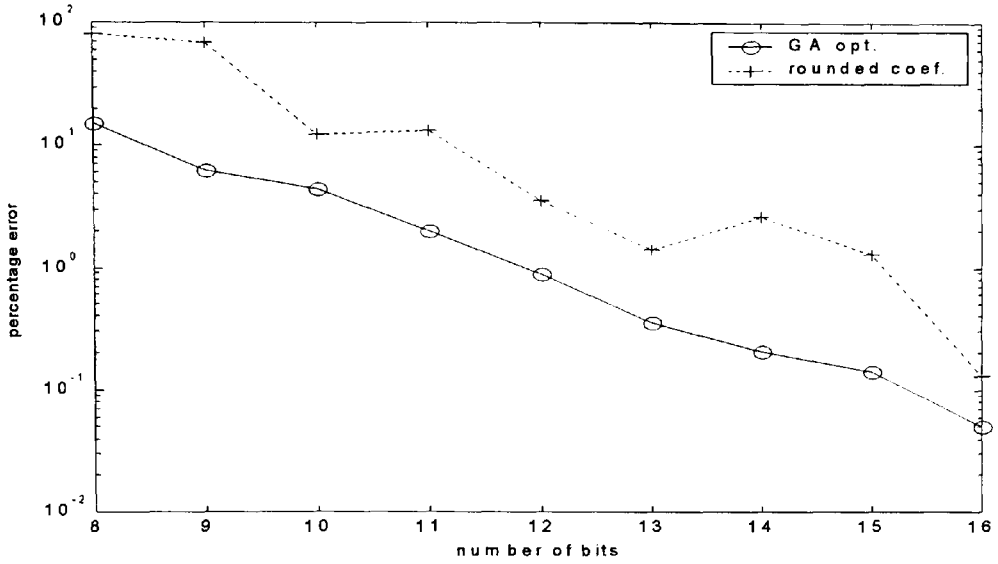

Figure 3.20 Magnitude response percentage error against number of bits of a second order cascade form 8th order high-pass filter with 'Infinity' norm.

3.4 Simple hill climber techniques and exhaustive search

To test the robustness and accuracy of the GA optimised results, the methods of simple hill climber algorithms such as the steepest ascent (SAHC) and the nearest ascent (NAHC) were applied to a selection of filters shown in Table 3.3 for the second-order cascade form structure of IIR filters. Random sampling tests for the search space as used for the GA optimisation was also conducted. Furthermore, for the selection of low order filters, an exhaustive search was conducted over a matching search space. The hill climber algorithms for this search are identical to those used in Chapter 2 (section 2.5) for FIR filters. The flow chart shown in Figure 2.14 describes the hill climber algorithm used for this application. In order to maintain parity with the GA optimisation, approximately the same number of objective function evaluations were performed for the hill climber methods. The hill climber performed a maximum of 120 objective function evaluations i.e. 12 evaluations (for filter length 12) for each loop running a maximum of 10 times. 20 runs of each algorithm, starting with a different randomly generated

seed thus generates a maximum of 2400 evaluations. The GA performs a maximum of 2120 evaluations. An important observation for the application of hill climber algorithms is that the search space for optimisation can extend beyond the range of +1 or -1 of the rounded values for each coefficient. This outcome is implicit in the evolutionary nature of the algorithms since mutation of the coefficient value occurs for each iteration. In this respect, there is a subtle difference when compared with the GA optimisation because the search space for GA is confined to +1 and -1 of the rounded coefficient values for the results obtained in this study. The hill climbers are thus subjected to a wider search space that may or may not be advantageous to the optimisation process. There is a possibility of obtaining a superior solution when compared to the GA method however, there is also a danger for the search to move towards areas of inferior or local minima solutions. The results of SAHC, NAHC, the random sampling and exhaustive search for a selection of the IIR filters are shown in Table 3.4 and the filter coefficients are shown in Table 3.5. The results shown with an asterisk (*) are the ones for which the search space has deviated greater than +1 or -1 of the rounded coefficient values. Note also that the exhaustive search was confined to deviation of +1, 0 or −1 of the rounded coefficients.

Table 3.4

GA and hill climber optimisation results of summed magnitude error of Equation 3.7 over 500 frequency points for second order cascade form IIR filters.

Filter	Exh. Sch	Random	SAHC	NAHC	GA
1) NN/SOS/LP4/5	6.1447	6.1447	9.3285	30.3495	6.1447
2) NN/SOS/LP4/8	0.9172	0.9253	0.8200*	2.2104	0.9172
3) IN/SOS/LP4/8	0.7966	0.7966	0.8812*	1.7215*	1.0973
4) IN/SOS/HP4/5	9.2286	11.5684	11.0276	21.6502	10.3195
5) NN/SOS/HP4/8	1.1778	1.4768	1.0779*	3.9608*	1.4458
6) NN/SOS/BP4/5	14.9344	22.0861	14.9344	52.2511	14.9344
7) IN/SOS/BP4/8	3.1483	3.1483	3.2204	5.0610	3.1483

Exh. Sch = Exhaustive search
Randon=random sample
SAHC=steepest ascent hill climber
NAHC=nearest ascent hill climber
* indicates search space exceeded +1 and/or −1 of rounded coefficient values

Table 3.5 The filter coefficients of selected filters.
* indicates search space exceeded +1 and/or −1 of rounded coefficient values

Filter 1: NN/SOS/LP4/5						
Random	1	3	2	15	-9	4
	14	12	15	15	0	12
SAHC	2	3	1	15	-8	4
	15	13	15	15	0	11
NAHC	3	1	2	15	-9	5
	14	14	14	15	-1	13
Filter 2: NN/SOS/LP4/8						
Random	13	22	14	127	-78	39
	127	114	127	127	0	103
SAHC	13	22	13	127	-79	39
	128*	114	127	127	-1	103
NAHC	5	22	12	127	-79	39
	125	115	127	127	0	103
Filter 3: IN/SOS/LP4/8						
Random	12	21	13	123	-76	38
	127	112	127	123	0	100
SAHC	13	21	12	123	-74	37
	129*	113	128	123	0	99
NAHC	14*	20	12	123	-76	38
	125*	116*	126	123	0	100
Filter 4: IN/SOS/HP4/5						
Exh. Sch	1	-2	0	15	15	6
	14	-4	15	15	8	12
Random	1	-1	1	15	15	6
	15	-5	15	15	9	13
SAHC	1	-1	1	15	14	6
	16	-6	14	15	9	12
NAHC	2	-1	-1	15	15	5
	15	-5	13	15	9	10
Filter 5: NN/SOS/HP4/8						
Exh. Sch	8	-12	8	127	118	48
	127	-38	126	127	71	104
Rand	8	-12	8	127	119	49
	127	-38	127	127	70	104
SAHC	8	-12	8	127	119	49
	126	-37*	126	127	71	104
NAHC	11*	-10*	7	127	121*	51*
	126	-39	125*	127	71	104
Filter 6: NN/SOS/BP4/5						
Random	2	4	2	8	7	4
	8	-15	8	8	-8	5
SAHC	2	4	2	8	7	4
	7	-15	8	8	-7	4
NAHC	4	5	2	8	5	3
	7	-12	6	8	-7	3
Filter 7: IN/SOS/BP4/8						
Random	11	23	12	58	51	33
	64	-127	64	58	-52	33
SAHC	12	23	11	58	51	33
	65	-128	64	58	-52	33
NAHC	14	25	12	58	53	33
	61	-123	65	58	-53	33

3.5 Discussion of results

As no quantifiable results of example form and structure of IIR filters and their FWL optimised coefficients are available in literature, an extensive range of such filters have been arbitrarily selected and used in this study. These cover both the direct form and the second order cascade form structures. The investigation of GA optimisation for each type of example filters was conducted and the results are reported. Table 3.2 shows the GA optimised results compared with the simply rounded values coefficients for the direct form IIR filters. The values listed are the summation of the absolute magnitude error between the exact and the approximate frequency response of the filter over 500 equidistant frequency points covering the full range of frequency spectrum from 0 to π radians. It must be recognised that for stability of IIR filters, all poles of the transfer function must lie inside the unit circle of the 'z' plane. This condition is embedded in the GA optimisation code thus leading to results that assure stable filter design. If, however, all the zeros of the transfer function are also either inside or on the unit circle then the filter generates the least phase form and is thus called a minimum phase filter. The GA optimisation was conducted for the case of minimum phase (MP) and non-minimum phase (NMP) conditions of IIR direct form filters in case of the example filter specifications. The results of GA optimisation listed in Table 3.2 show a significant improvement over the simply rounded coefficient values. Furthermore, the non-minimum phase optimised results show consistently improved performance of the magnitude response when compared with the minimum phase optimised results. In either case, the optimised results show significant improvement when compared with the results using the simply rounded coefficient values.

The same set of filters as used above in the direct form implementation of IIR filters were also GA optimised in the form of second order cascade structures. The results for this form of IIR

filters are shown in Table 3.3. Two scaling options were used i.e. 'none norm' (NN) and the L_∞ - 'infinity norm' (IN). Note that the 'UP' ordering of pole-zero pairs of the cascade sections and the L_∞-norm scaling minimises the probability of overflow error in the realisation of the IIR filters. The GA optimised results using the 'none-norm' option generate largely improved results when compared with the 'infinity-norm' optimised results. However, in either case the optimised results show significant improvement when compared with the results using the simply rounded coefficient values. Furthermore, when compared with the optimised results of the direct form IIR structure shown in Table 3.2, the second-order section cascade structure results show significantly performance clearly demonstrating high immunity of such structures to FWL coefficient effects. A complete listing of the optimised coefficient values for the direct form and the second order cascade form are shown in Appendices C1.2 and C2.2 respectively.

Further tests were conducted on a selection of IIR filters of the second order cascade form structures using the simple hill climber techniques, random sampling and exhaustive search. The results of these tests are shown in Table 3.4. Once again, the GA optimised results are seen to be consistently good. However, for some filters such as the Filter2: NN/SOS/LP4/8, Filter3: IN/SOS/LP4/8 and the Filter 5: NN/SOS/HP4/8, the steepest ascent hill climber method has generated superior results. This is significant since the search space for these algorithms can intrinsically extend beyond the +1 and/or −1 of the rounded coefficient values. The GA search space, however, is restricted to +1 or −1 of the rounded coefficients. This offers credibility to the simple hill climber technique and complements the GA optimisation to search for superior solutions for the application considered here.

3.6 Summary of Chapter 3

The specific problem of finite word length coefficients in the realisation of IIR filters has been considered here. The purposeful aim is to use the procedures of genetic algorithms to optimise the frequency response in comparison to the exact filter response. Quantifiable metrics is based on the calculation of magnitude response error using Equation 3.7 over the full frequency range. Comparison is drawn between the simply rounded coefficient results and those obtained using the GA optimised coefficients. Two types of IIR filters have been considered i.e. the direct form and the second-order cascade form. In each case further sub-divisions have been considered. For direct form filters, two types i.e. minimum phase and non-minimum phase realisations have been considered and for second order cascade form the 'UP' ordering of poles in conjunction with firstly, 'Infinity' norm and then 'None' norm have been considered.

The result, as shown in Tables 3.2 and 3.3 show a distinct improvement of GA optimised results in comparison to the simply rounded coefficient results. A general comparison over a number of bits using the percentage error metric given by Equation 3.16 for selected filters have been shown in Figure 3.7 (for direct form) and in Figures 3.16 and 3.17 (for second order cascade form). In each of these figures, the GA optimised results show a distinct improvement, when compared to rounded coefficient results, over the whole range of bits used for coefficient representation.

The general conclusion of this part of the study leads to the observation that the second-order section realisation of IIR filters generates good results over a range of filter types. However, the accuracy of results is dependent on the number of bits used for coefficient representation and the type of implementation (see Table 3.3). The stability-check criteria have been applied to all the GA optimised results and a listing of the coefficient values have been included (Appendices

C1.2 and C2.2). The GA optimisation of IIR filters as discussed and investigated in this chapter is clearly useful in the implementation of FWL coefficient structures for real-time applications. This study will be applied to the case of multirate quadrature mirror filter banks using IIR filters that will be investigated in the next chapter.

The major contributions of this part of the study are the following.

- Real integer-valued genetic algorithm codes have been developed for the optimisation of the finite word length constrained coefficients of IIR digital filters. The direct form and the second order cascade form structures have been considered. Further distinctions drawn are in terms of minimal and non-minimal phase IIR direct form filters and the use of 'Infinity norm' and 'none norm' for the case of IIR second order cascade form structures.
- In order to establish a basis for comparative study, a number of band select filters have been defined as seen in Table 3.1. The GA optimised results for different filter orders and number of bits representing the coefficient values are seen to be vastly superior when compared with the simply rounded coefficient value results. These results are shown in Tables 3.2 and 3.3 and the actual coefficient values are listed in Appendices C1.2 and C2.2.

Chapter 4: Optimisation and real-time implementation of a class of multirate quadrature mirror filter bank

Overview of Chapter 4: This chapter starts with an introductory section on the 2-channel multirate quadrature mirror filter (QMF) bank. Some theoretical issues regarding reconstruction errors and conditions for perfect reconstruction (PR) of the input signal are considered. The main emphasis is on the optimisation of a new design of a perfect reconstruction QMF bank using IIR filters. The GA optimisation of the initial design of the QMF bank and of the IIR filters using finite word length coefficients is investigated and reported. The optimised results are then applied to a real time digital signal processing kit. Finally, some test results for data compression achievable using different values of encoded bits are included.

4.1 Introduction

The issues of finite word length errors and their optimisation for real-time realisation of digital filters as covered in Chapters 2 and 3, has important application in the field of multirate filter banks. This is mainly because the design of multirate filter banks is based on the use of FIR and/or IIR digital filters. The real-time applications of multirate banks, such as sub-band coding of telephony speech signals and data compression, is based on using finite word length form of the input signal, digital filter coefficients and arithmetic operations on fixed-point digital signal processing devices. The work developed in Chapters 2 and 3 is extended to the case of real-time realisation of multirate filter banks that is considered in this Chapter.

Multirate processing of digital signals is involved with variable rate sampling at different stages of a system often resulting in efficient processing of signals. The main areas of application of

multirate signal processing include digital audio systems, speech and image processing, transmultiplexers, sub-band coding and signal data compression. Multirate filter banks are of a specific structure with applications in spectrum analysis and sub-band coding of speech and image signals [Vaidyanathan, 1990]. An example of a specific form of a two-channel multirate structure shown in Figure 4.1 is commonly referred to as a quadrature mirror filter (QMF) bank. This is due to the power complementary frequency response of the low pass and high pass filters used.

A requirement for perfect reconstruction (PR) of the input signal through a 2-channel filter bank as shown in Figure 4.1 is the cancellation of amplitude, phase and aliasing distortions of the output signal. Theoretical methods of achieving PR are well established [Vaidyanathan, 1993]. However, obtaining good sub-band filters of high order and minimising reconstruction errors are key elements for the specified design and implementation of real systems. Most methods for designing a QMF bank use FIR analysis and synthesis filters. The linear phase characteristic of FIR filters result in the elimination of phase distortion and with appropriate choice of the synthesis filters, aliasing error is also eliminated. Finally, the amplitude distortion is minimised by an appropriate optimisation procedure to give a near-perfect reconstruction of the input signal. However, a large number of coefficients are required for high sub-band frequency separation for the implementation of FIR filters that makes their use somewhat inefficient. For this reason IIR filters are preferred since fewer coefficients are required for similar frequency response specifications although stability checks are required and in general, non-linear phase response of IIR filters can cause undesirable effects for specific applications such as image processing.

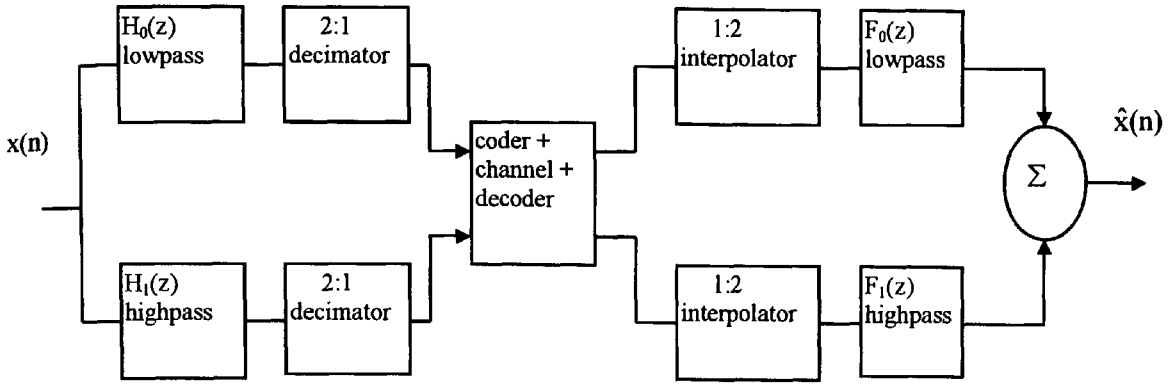

Figure 4.1 A two channel quadrature mirror filter bank.

Most types of IIR filter banks proposed in literature either generate non-causal filters or they do not achieve perfect reconstruction [Vaidyanathan, 1993], [Vetterli and Kovacenic, 1995]. A broad class of IIR QMF banks have been extensively studied resulting in efficient all-pass based realisations [Vaidyanathan et al 1987]. However, phase distortion in these structures remains a problem due to the intrinsic non-linear phase response of IIR filters. Some form of all-pass equalisation must be used to overcome the phase distortion. Such an equalisation process is complex and is not efficient for real-time applications. Other methods reported in literature include mixed mode, FIR/IIR implementation. The H_∞ optimisation method pre-specifies the analysis filter (FIR or IIR) for efficient coding of the transmitted signal and the IIR synthesis filter bank is designed based on H_∞ optimisation [Chen and Francis, 1995]. Work reported in Zhu et al [1998], uses an all-pass IIR filter for the analysis part of the QMF bank and a non-linear phase FIR filter that is designed using a weighted least-square (WLS) algorithm to obtain the synthesis filter.

Yet another class of IIR QMF bank design is based on the application of transformations. This category includes the use of the McClellan transformation that was originally used for

transforming zero-phase 1-D FIR filter to 2-D FIR filter [McClellan, 1973]. An equivalent generalised transformation of McClellan has been used for designing causal IIR analysis and synthesis filters and is shown to be flexible in being able to use a large class of transformation functions [Tay and Kingsbury, 1996]. The work due to Basu et al [1995] is concerned mainly with the theoretical issues of complete parameterisation of the filter bank system and no design examples for implementation are given. The work reported in Phoong et al [1995] uses a polyphase representation of the 2-channel QMF bank and the design of all analysis and synthesis filters is reduced to the design of a single transfer function. However, the design procedure is not direct and involves substantial constraints. A minimax design approach of IIR QMF banks has been reported recently [Lee and Niu, 2001]. In this work, the frequency response is optimised in the L_∞ minimax sense; however, the design of an IIR low pass prototype filter is based on heuristic initial assumption of coefficient values that cannot always assure optimal minimisation of the error function.

The multirate filter investigated in this chapter is based on a method of designing a 2-channel perfect reconstruction IIR filter bank using the transformation of variables technique. This technique was originally developed for designing multidimensional FIR filter banks [Tay and Kingsbury, 1993] but was later extended to the case of IIR filter banks [Tay and Kingsbury, 1996], [Tay, 1998]. The 'transformation of variables' technique involves the use of small prototype filters and transformation of their variables by applying a transformation function. The transformation function uses a number of parameters that must be determined and optimised for appropriate frequency response of sub-band filter transfer functions. Although such a design method is simple and flexible, some heuristic assessment, based on trial-and-error procedure for the variables is used in order to obtain desirable results. Even so, there is no assurance that a near-optimal result has been obtained. It is this feature that inspired the use of genetic algorithms for locating quasi-optimal values, both in the design stage of the 2-channel

filter bank and further optimisation using finite word length constraints for real time implementation. The major attraction for using the 'transformation of variables' technique is the design of causal stable IIR filters generating good frequency band separation and satisfying the perfect reconstruction condition. Furthermore, this technique is flexible in being able to use a large class of transformation functions thus leading to a number of options for the design implementation in real time.

4.2 Errors in the QMF bank

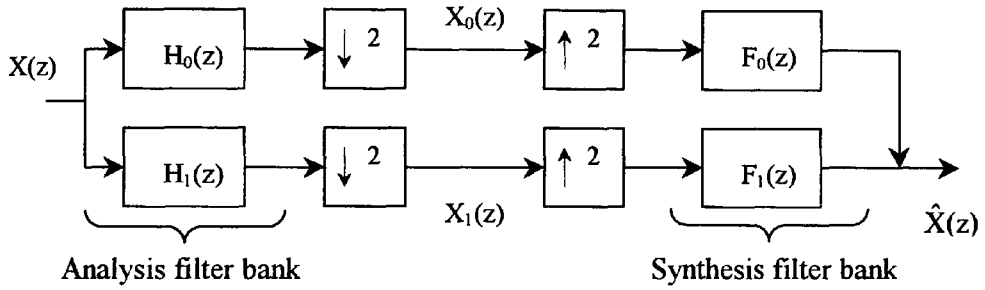

Figure 4.2 Sub-band coding of a QMF bank.

The analysis filter bank shown in Figure 4.2 decomposes the input signal X(z) into sub-band signals $X_0(z)$ and $X_1(z)$. This is followed by the synthesis filter bank that reconstructs the signal $\hat{X}(z)$ from the sub-band signals. For real signals, the power of the original signal may not be equally distributed over the sub-bands. This property can be exploited constructively and can lead to coding gain if the sub-band signals are independently coded instead of the original signal [Jayant and Noll, 1984]. The coded signal can be transmitted and then decoded at the receiver where the original signal is reconstructed. The analysis and synthesis filters must be designed such that $\hat{X}(z)$ is as close as possible or even exactly the same as the input signal X(z). The mathematical relationship between the signals and the filters as shown in Figure 4.2 [Fliege, 1994] is given by

$$\hat{X}(z) = \begin{bmatrix} F_0(z) & F_1(z) \end{bmatrix} \frac{1}{2} \begin{bmatrix} H_0(z) & H_0(-z) \\ H_1(z) & H_1(-z) \end{bmatrix} \begin{bmatrix} X(z) \\ X(-z) \end{bmatrix} \quad 4.1$$

Equation 4.1 can be expanded to

$$\hat{X}(z) = \tfrac{1}{2}X(z)[F_0(z)H_0(z) + F_1(z)H_1(z)] + \tfrac{1}{2}X(-z)[F_0(z)H_0(-z) + F_1(z)H_1(-z)] \quad 4.2$$

The second term of Equation 4.2 represents the aliasing caused by the overlapping of frequency responses due to sampling rate alteration and can be expressed as:

$$A(z) = \tfrac{1}{2}[F_0(z)H_0(-z) + F_1(z)H_1(-z)] \quad 4.3$$

An alias-free realisation requires A(z) to be zero and the following choice of synthesis filters achieves this condition i.e.

$$F_0(z) = H_1(-z) \quad \text{and} \quad F_1(z) = -H_0(-z) \quad 4.4$$

The first term of Equation 4.2 describes the distortion transfer function T(z) of the alias free filter bank i.e.

$$T(z) = \tfrac{1}{2}[F_0(z)H_0(z) + F_1(z)H_1(z)] \quad 4.5$$

By using the property of Equation 4.4 then

$$T(z) = \tfrac{1}{2}[H_0(z)H_1(-z) - H_1(z)H_0(-z)]$$

and the output of the alias-free QMF bank is given by

$$\hat{X}(z) = T(z)X(z) \quad 4.6$$

If T(z) is an all-pass function then $|T(e^{j\omega})| = d \neq 0$ for all ω and the QMF bank is said to be magnitude preserving. If T(z) has linear phase then $\arg[T(e^{j\omega})] = a\omega + b$ and the filter is said to be phase preserving. In order for a QMF bank to be a perfect reconstruction (PR) QMF bank then it must be alias-free, magnitude and phase preserving.

4.3 Design procedure using transformation of variables method

Several techniques for the design of a two-channel QMF banks exist [Vaidyanathan, 1993], [Mitra, 1998]. The main aim of the design is to develop conditions for the analysis and

synthesis digital filters so that the magnitude and phase response errors and the aliasing error are all minimised such that the perfect reconstruction characteristic of the QMF bank is closely satisfied. The problems of aliasing error, amplitude and phase distortions can cause degradation of the input signal thus a compromise solution must be found that gives an optimal condition for perfect reconstruction. Some of these errors are intrinsically eliminated due to the nature of the selected filter types and the remainder errors are minimised based on optimisation techniques. Further considerations are concerning computational overheads and complexity of filter structure in terms of throughput, power dissipation and other related costs.

The design method considered here is the transformation of variables technique proposed by Tay [1998]. This technique generates IIR filters with good frequency band separation that are casual and stable and can achieve perfect reconstruction. The basis of this method of design lies in the use of a set of small prototype filters and the transformation of their variables using a transformation function. A large range of transformation functions can be used each of which consist of a set of parameters that can be optimised to give the desired filter characteristics. This affords flexibility to the designer for 'fine-tuning' the characteristics and the final outcome, with relative ease.

The transformation of variables technique was initially developed for designing multidimensional linear phase FIR filter banks with applications to image processing [Tay and Kingsbury, 1993]. Further development of this technique is in designing IIR filter banks both in 1-D and 2-D [Tay and Kingsbury, 1996], [Tay, 1998]. The low pass filters $H_0(z)$ and $F_0(z)$ (see Figure 4.2) are derived from prototype filters $H_T(Z)$ and $F_T(Z)$ respectively that are both functions of a polynomial in Z. The transformation applied is given by $Z=M(z)$ that satisfies the condition $M(z) = - M(-z)$.

The low pass filters are thus given by

$$H_0(z) = H_T(M(z)) \qquad \text{and} \qquad F_0(z) = F_T(M(z)) \tag{4.7}$$

and the high pass filters are given by

$$H_1(z) = z^{-K}F_0(-z) \qquad \text{and} \qquad F_1(z) = z^{K}H_0(-z) \tag{4.8}$$

where K is an odd integer.

The design of the prototype filters is subject to the constraint that

$$H_T(Z)\,F_T(Z) + H_T(-Z)\,F_T(-Z) = 2 \tag{4.9}$$

The condition of Equation 4.8 reduces the input/output relationship of the QMF bank given by Equation 4.2 to an alias-free form, thus

$$\hat{X}(z) = \tfrac{1}{2}X(z)[F_0(z)H_0(z) + F_1(z)H_1(z)] \tag{4.10}$$

Using Equations 4.7, 4.8 and 4.9 the Equation 4.10 reduces to $\hat{X}(z) = X(z)$ i.e. perfect reconstruction.

A number of prototype filters have been considered by Tay and Kingsbury [1993] and their properties analysed. The most promising for sub-band filter banks is the pair obtained for the modified Lagrange half band filter given by

$$\left.\begin{aligned} H_T(Z) &= -\tfrac{1}{4}(Z+1)(Z-3) \\ F_T(Z) &= -\tfrac{1}{12}[(Z+1)(Z^2+Z-8)] \end{aligned}\right\} \tag{4.11}$$

Furthermore, only rational transformations are considered in this work that generate IIR filters. The design of the prototype filters is based on the value of $Z=M(e^{j\omega})$ moving on a complex contour 'C' given by

$$C = \{ Z: Z = M(e^{j\omega});\ -\pi \le \omega \le \pi \} \tag{4.12}$$

The contour that gives most flexibility and thus offers a range of possible design options is an elliptical contour given by

$$C_e(b) = \{ Z: Z = \cos(\theta) + jb\sin(\theta);\ -\pi \le \theta \le \pi \} \tag{4.13}$$

where $0 \le b \le 1$

A set of contours can be generated that lie between the two extreme cases by changing the value of 'b'. These contours can be approximated by a transformation function for Z, of the form

$$Z = M(z) = kz^{P}\prod_{i=1}^{Pr}\frac{c_i z^2 + 1}{z^2 + d_i}\prod_{i=1}^{Pc}\frac{z^4 r_i^2 - 2r_i\cos\varphi_i z^2 + 1}{z^4 - 2p_i\cos\psi_i z^2 + p_i^2} \qquad 4.14$$

where $P = 2P_r + 4P_c - 1$ and

$$k = \prod_{i=1}^{Pr}\frac{1+d_i}{1+c_i}\prod_{i=1}^{Pc}\frac{p_i^2 - 2p_i\cos\psi_i + 1}{r_i^2 - 2r_i\cos\varphi_i + 1} \qquad 4.15$$

the normalisation factor k assures that Z = 1 when θ is zero.

As mentioned above, the sub-band filters obtained by using this method are casual and stable (see Tay [1998] for proof). There are several parameters to be designed for the higher order transformations and a trial and error method is not practical, hence some form of optimisation method is more suitable. The objective function to be minimised is a function of the parameters **c, d, r, φ, p, ψ** of the transformation function M(z) and is given by

$$\text{Obj\_fn} = \sum_{m=1}^{L} |M(e^{j\omega_m}) - M_i(\omega_m)|^2 \; W(\omega_m) \qquad 4.16$$

note that **c, d, r, φ, p, ψ** are the design parameter vectors e.g. $\mathbf{c} = [c_1,\ldots,c_{Pr}]$. Also, $M(e^{j\omega_m})$ is the frequency response of M(z) at ω_m and $M_i(\omega_m)$ is the desired frequency response of M(z) at ω_m. $M_i(\omega)$ represents the desired complex contour $C_e(b)$. Only the passband frequencies need to be considered in the objective function as $M(e^{j\omega})$ is conjugate anti-symmetric about the frequency π/2 (refer to [Tay, 1998] for proof). The summation is over a set of frequencies ω_1, $\omega_2, \ldots, \omega_L$ where $\omega_1 = 0$ and $\omega_L = \pi/2$ representing the passband edge. The idealised function M_i is given by

$$M_i(\omega) = \cos\theta(\omega) - j\sin\theta(\omega) \qquad 4.17$$

where $\theta(\omega) = K\omega^n$ for $0 \le \omega \le \pi/2$ and K is a normalisation factor= $(\pi/2)^{1-n}$.

The value of 'n' defines the roll-off requirement to be achieved by the optimisation process. A high value of 'n' will result in a filter with sharper roll-off. $W(\omega_m)$ in Equation 4.16 is the weighting function and can have values of

$W(\omega_m) = m$, for a slow roll-off requirement i.e. positive linear weighting; or

$W(\omega_m) = L+1-m$, for a sharp roll-off requirement i.e. negative linear weighting.

It must be mentioned that a sharp roll-off tends to increase the ripple, whilst a slow roll-off tends to reduce it. For large values of n i.e. when $n \rightarrow \infty$, then

$$M_i(\omega) \rightarrow \begin{cases} 1 \text{ for } 0 \leq \omega \leq \pi/2 \text{ (passband)} \\ \\ -1 \text{ for } \pi/2 \leq \omega \leq \pi \text{ (stopband)} \end{cases} \qquad 4.18$$

this is a typical idealised 'brick-wall' type frequency response.

The design problem of the prototype filters is thus reduced to minimising the objective function Equation 4.16 subject to the constraints

$-1 \leq d_i < 1$ and $0 < p_i < 1$ for stability; and 4.19

$M(z=e^{j\pi/2}) = -jb$ 4.20

to ensure that the complex contour passes through the point (0, -jb).

From the transformation function of Equation 4.14, it can be seen that the simplest function is obtained when $P_r = 1$ and $P_c = 0$. This leads to essentially only one parameter to design for the required response. A trial-and-error approach is then easily applied to arrive at the desired response. However, for higher order transformations, there are several parameters involved and a simple trial-and-error method is not practical. For this, a more comprehensive optimisation technique is required. Note that for the two cases of the transformation function used in the design examples considered in this work, the number of parameters required to be optimised can be reduced [Tay, 1998]. For example for $P_r=2$ and $P_c=0$, then

$$c_1 = \frac{(1+d_1)(1+d_2)(1-c_2) - b(1-d_1)(1-d_2)(1+c_2)}{(1+d_1)(1+d_2)(1-c_2) + b(1-d_1)(1-d_2)(1+c_2)} \qquad 4.21$$

and for $P_r = P_c = 1$, then

$$c = \frac{F(1+d) - b(1-d)}{F(1+d) + b(1-d)} \qquad 4.22$$

where

$$F = \left(\frac{p^2 - 2p\cos\psi + 1}{r^2 - 2r\cos\varphi + 1}\right)\left(\frac{r^2 + 2r\cos\varphi + 1}{p^2 + 2p\cos\psi + 1}\right)$$

The work reported in Tay [1998] uses a trial set of 'seed' parameter values for the constrained optimisation algorithm function constr.m of MATLAB to obtain a converged solution. A global optimal solution is not assured with such gradient-based methods so a number of trial 'seed' parameters must be used to obtain desirable results. This is a major limitation of the 'transformation of variables' technique especially for higher order transforms and has led to the motivation for the work that is covered in this chapter. A genetic algorithm approach is developed to search for global minima. Furthermore, this work is extended towards obtaining finite word length, causal stable IIR filters through a second stage GA optimisation procedure for real-time applications as developed in Chapter 3.

4.4 Genetic algorithm implementation procedure and methodology

The major limitation of the optimisation process of the parameters of the transformation function covered by Tay [1998] is the use of the Matlab function 'constr.m' that is based on the constrained non-linear i.e. sequential quadratic programming (SQP) method. This process works well but is highly susceptible in converging to a local minima point. A global optimal

solution is not assured so a number of starting seed values must be tried. This is clearly restrictive and the problem is further compounded for higher order transformation functions that consist of larger number of parameters. GA optimisation is thus a good option in such applications due to its intrinsic search capability over a much wider landscape of the objective function.

In order to obtain the final optimised design based on the transformation of variables technique of the QMF bank for the FWL constrained real-time realisation, a number of steps must be followed. In brief these are

1. Select the order of the transformation function Z=M(z) (see Equation 4.14) to be optimised and define the parameters of the idealised function given by Equation 4.17 i.e. $M_i(\omega) = \cos\theta(\omega) - j\sin\theta(\omega)$, where $\theta(\omega)=K\omega^n$ for $0 \leq \omega \leq \pi/2$ and K is a normalisation factor = $(\pi/2)^{1-n}$. Note that for $n \rightarrow \infty$ then $M_i(\omega)$ generates the standard ideal brick-wall frequency response.

2. The transformation function design parameters given by c_i, d_i, r_i, ϕ_i, p_i, φ_i are optimised using the objective function given by Equation 4.16 that is minimised i.e. $\text{Obj\_fn}=\sum_{m=1}^{L} | M(e^{j\omega_m}) - M_i(\omega_m) |^2 \; W(\omega_m)$ where $W(\omega_m)$ is a weighting factor. A pseudo code for this step of design optimisation using GAs is shown in Figures 4.3 and 4.4 for the main code and the objective function code respectively.

3. Use the prototype filters of Equation 4.11 and the optimised transformation function from step 2 to derive the transfer function coefficients of the digital filters using Equations 4.7 and 4.8. This step generates the IIR analysis and synthesis filters with

high precision coefficients and the design stage of the QMF bank is completed. A pseudo Matlab code for this step is shown in Figure 4.5.

4. This step is the second stage of the process where the high precision coefficients of the analysis and synthesis filters are converted to the finite word length form and then optimised using the GA techniques developed in Chapter 3.

5. The final step is the real-time implementation of the optimised QMF bank on a digital signal processing hardware system.

It must be recognised that although the design stage of the QMF bank in this study generates IIR filters, there exist several other design methods that involve the use of FIR filters. For such applications the FWL optimised techniques developed in Chapter 2 can then be applied. An important observation in the above steps for final realisation of the QMF bank is that the optimisation of design is entirely independent of the FWL coefficient optimisation of the IIR filters. The two stages are thus considered separately since entirely different constraints and issues are involved with their optimisation processes. A combined code for the two stages for which the entire process is linked in a sequential form, however, is trivial.

The genetic algorithm used here for the design of the QMF bank is identical to the generic form explained in section I.4. This is a MATLAB based algorithm developed originally for control systems applications [Chipperfield et al, 1993]. The main GA code has been adapted for the application in this work and new functions have been written for working out the error objective function. Figure 4.6 shows a complete flow chart procedure for obtaining a filter realisation for real-time implementation of the 2-channel sub-band filter banks. The specific steps followed for the design stage of the QMF bank GA optimisation are

1) Define the GA parameters

Number of individuals = 200, Number of generations = 100, Generation gap = 0.9
Reinsertion rate = 0.7, Mutation rate = 0.2, Number of frequency points sampled L = 100

2) Create population set of individuals

The starting set of parameter values r_i, ϕ_i, φ_i are initially assumed within the bounds of –2 and +2 although stability bounds for d_i and p_i are strictly applied as defined in Equation 4.19. The bounded parameter values are described in a matrix 'FieldDR' and an initial population set consisting of random real-valued individuals is created within the bounds specified in FieldDR matrix. The function **crtrp** of the GA Matlab toolbox is used for this purpose.

3) The Objective function evaluation

The main purpose of the optimisation process here is to minimise the objective function with the specific aim of obtaining an approximated frequency response of the transformation function that is as close as possible to the desired response. The objective function used here is given by Equation 4.16 i.e.

$$\text{Obj\_fn} = \sum_{m=1}^{L} | M(e^{j\omega_m}) - M_j(\omega_m) |^2 \; W(\omega_m)$$

Where $W(\omega_m)$ is a weighting factor and it varies for different design examples considered.

4) Fitness value and ranking

The Matlab based **ranking** function of the GA toolbox ranks the individuals according to their objective function values 'Obj_fn' and returns a column vector consisting of the corresponding fitness value 'FitnV' of the individuals. This function performs a linear ranking with a selective pressure (SP) of 2 [Whitley, 1998]. The fitness value assigned to the individuals is calculated according to the formula given by Equation 1.1 Chapter 1.

5) Selection of individuals for breeding

The high-level function for selection of individuals from the population set and returning the selected individuals in a new population is performed by the **select** function. The low-level

selection function **sus** is called by the **select** function. The **sus** function is based on a form of stochastic sampling method and is implemented by obtaining a cumulative sum of the fitness vector 'FitnV' and generating a set of equally spaced numbers between 0 and Σ(FitnV) [Baker, 1987]. The probability of an individual being selected is given by Equation 1.2 in Chapter 1.

6) Recombining individuals – crossover

The crossover function is also performed in two stages. The high-level function is **recombin** that calls the low-level function **recdis**. The **recdis** function is a discrete recombination function. The mating process is performed between pairs of rows. The **recdis** function first generates an internal mask table that determines which parents contribute which variables to the offspring. On the basis of the randomly generated mask table, the variable values are exchanged between the individuals and return a new population after mating.

7) Mutation

The **mutbga** function of the Matlab GA toolbox takes real-valued population, mutates each variable with given probability and returns the population after mutation. The **mutbga** function produces firstly a random internal mask table that determines which variables will mutate and also the sign for the step size. A second internal table generates the normalised mutation step size. The mutated variable is worked out as a function of the original variable and the step size [Muhlenbein and Schlierkamp-Voosen, 1993].

8) Reinsert offspring into new population

The new population set generated after crossover is subjected to the objective function evaluation of each new individual. On the basis of their fitness, the offspring are selected for reinsertion using the **reins** function into the new population. The objective function values are then copied to the reinserted offspring and the GA loop is then repeated for the next generation.

A pseudo GA code for the transformation function filter optimisation and the objective function code are shown in Figures 4.3 and 4.4 respectively.

```
% Pseudo GA optimisation of transformation function parameter values

% GA and transformation fn. parameters
Qw=((pi/2)^(1-n))*w.^n;     % weighting fn.
Mi=cos(Qw)-j*b*sin(Qw);     % idealised filter

% Built field descriptor
FieldDR=[-1 -2 -2 0 -2;1 2 2 1 2];

% Initialise population
Chrom=crtrp(NIND,FieldDR);

% Evaluate initial population
ObjV=tvgaobj(Chrom,z,Mi,b,W);
gen=0;   %counter

% Generational loop
while gen < MAXGEN

      %Assign fitness values to entire population
   FitnV = ranking(ObjV);

      %Select individuals for breeding
   SelCh=select('sus', Chrom, FitnV, GGAP);

      %Recombine individuals (crossover)
   SelCh=recombin('recdis',SelCh);

      %Apply mutation
   SelCh=mutbga(SelCh,FieldDR,MutRate);

      %Evaluate offspring, call objective function
   ObjVSel=tvgaobj(SelCh,z,Mi,b,W);

      %Reinsert offspring into population
   [Chrom ObjV]=reins(Chrom,SelCh,1,[1 INSR],ObjV,ObjVSel);

      %Increment counter
   gen=gen+1
   [m,n]=min(ObjV);
   ObjV(n,1)
   OBJ(gen)=ObjV(n,1);
   if gen > SWOVT;   % start of creep code
   % creep code here
   ObjV=tvgaobj(Chrom,z,Mi,b,W);
   end

end
```

Figure 4.3 Pseudo GA optimisation of the transformation function parameter values.

```
% Objective function for tv_ga.m

function f=tvgaobj(Chrom,z,Mi,b,W);

% Optimisation function for one first-order and one
% second order transformation function.


F1=((p^2)-2*p*cos(psi)+1)/((r^2)-2*r*cos(phi)+1);
F2=((r^2)+2*r*cos(phi)+1)/((p^2)+2*p*cos(psi)+1);
F=F1*F2;
c=(F*(1+d)-b*(1-d))/(F*(1+d)+b*(1-d));

k1=((1+d)/(1+c));
k2=((p^2)-2*p*cos(psi)+1)/((r^2)-2*r*cos(phi)+1);
k=k1*k2;

% P = 2Pr+4Pc-1 = 5
P=5;

Mz1=(c*(z.^2)+1)./((z.^2)+d);
Mz2=((z.^4)*(r^2)-2*r*cos(phi)*(z.^2)+1)./((z.^4)-
2*p*cos(psi)*(z.^2)+(p^2));
Mz=k*(z.^P).*Mz1.*Mz2;

f(i,1)=sum(((abs(Mz-Mi)).^2).*W);

end;
```

Figure 4.4 Pseudo objective function code for GA optimisation of transformation function parameter values

```
% Deriving QMF bank IIR filter transfer functions using
% transformation of variable method utility (for Pr=Pc=1)
%
% [H0b,H0a,F0b,F0a]=tvfilter(c,d,r,phi,p,psi)
%
% H0b,F0b: numerator of digital filter
% H0a,F0a: denominator of digital filter
% design example 2

% optimised transformation function parameters
c=0.7491;d=0.5001;r=0.1918;phi=1.0318;p=0.1097;psi=1.0013;

% work out constant 'k'

% split the transformation function in a form Z=Z1*Z3/Z2*Z4
Z1=[k*c(1) 0 k 0 0 0 0 0]; Z2=[1 0 d(1)];
Z3=[r^2 0 -2*r*cos(phi) 0 1]; Z4=[1 0 -2*p*cos(psi) 0 p^2];

% polynomial multiplication is convolution fn. in Matlab
Zn=conv(Z1,Z3); Zd=conv(Z2,Z4);

% work out HT(Z) prototype filter eqn. 4.11
H1(1:5)=Zn(1:5); H1(6:12)=Zn(6:12)+Zd;
H2(1:5)=Zn(1:5); H2(6:12)=Zn(6:12)-3*Zd;
H4=conv(H1,H2); H5=4*conv(Zd,Zd);

% work out FT(Z) prototype filter eqn. 4.11
Z2n=conv(Zn,Zn); Z2d=conv(Zd,Zd);
F2n=conv(Zn,Zd); F1(1:5)=Zn(1:5);
F1(6:12)=Zn(6:12)+Zd; F2(1:5)=Z2n(1:5);
F2(6:23)=Z2n(6:23)+F2n;F2(11:23)=F2(11:23)-8*Z2d;
F3=conv(Zd,Z2d); F4=conv(F1,F2); F5=12*F3;

% numerator and denominator coefficients low pass analysis and
%synthesis filters
H0b=-H4/4; H0a=H5/4; F0b=-F4/12; F0a=F5/12;

% derive numerator and denominator coefficients of high pass analysis
%and synthesis filters i.e. H1b, H1a, F1b, and F1a using the eqn. 4.8
```

Figure 4.5 Pseudo code for deriving the analysis and synthesis IIR digital filter transfer functions using transformation of variable method utility (for Pr=Pc=1).

The GA optimisation of the transformation function parameter values is implemented to determine the individual with a minimum objective function value. After several trial runs of the first stage GA, it was observed that although good results were achievable, these did not always generate near optimal results when compared to those that were sometime obtainable using the gradient based optimisation function constr.m of the Matlab Optimisation toolbox. There is a dilemma here. The constr.m function generates good results but is dependent on the initial seed parameter values of the transformation function parameters. It was also observed that for some seed values, the constr.m function did not converge. This may be due to the characteristic of the transformation function landscape that represents fairly flat regions. Several seed values must be tried before a good desirable result is obtained. For this reason, a hybrid structure shown in Figure 4.6 is proposed, where a certain minimum objective function value is tested after the first stage GA. It was observed that this minimum value need not be too severe. Few trial runs of the GA give a good estimate of the minimum value thus generating a good set of parameter values. These are then used as seed values for the constr.m gradient based optimisation function to generate near optimal results. These results are an improvement to those reported by Tay [1998]. It must be recognised that a GA optimisation procedure alone cannot assure of a global optimal value. However, for this application, the parameter values

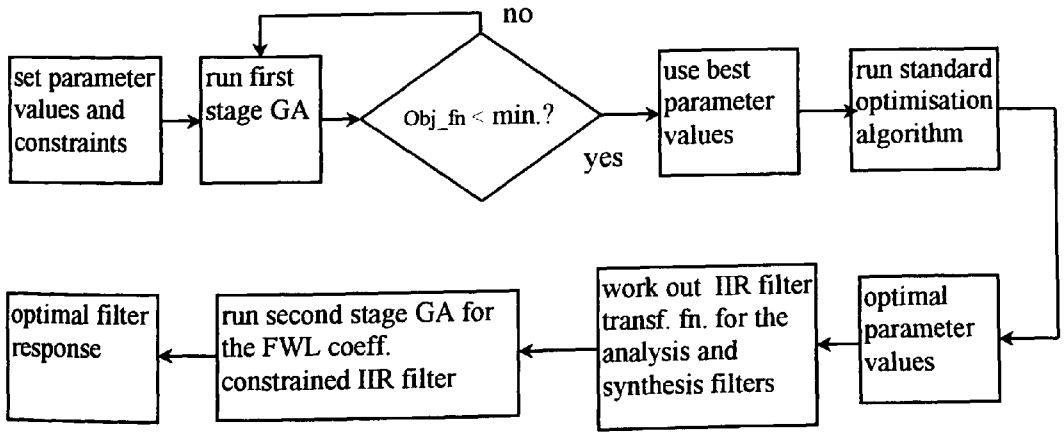

Figure 4.6 Flowchart for realisation of causal IIR filter using a two stage GA.

obtained from the first stage GA were found to be consistently good seed values for input to the constr.m optimisation function to generate desirable results. The optimal parameter values from the first stage are used to work out the coefficients of the causal IIR filter. The second stage GA is then applied to the FWL coefficient optimisation of the IIR filters as developed in Chapter 2. For this, the rounded coefficient values are obtained using

$$b_r = \text{round}\,[b_e\, 2^{N-1}] \qquad 4.23$$

where b_e is the high precision coefficient value and N is the number of bits.

The finite word length rounded coefficient values are used as a seed to generate a population set of new coefficients that are obtained by perturbing the integer values by +1, 0 or -1. A stability check is performed for every new individual and a high penalty is awarded to individuals not satisfying the stability requirements. Furthermore, the IIR filter can be implemented using either the direct form or a second-order cascade form structures depending on the design requirements. It is well recognised that a second-order cascade structure is significantly less sensitive to coefficient variations due to rounding effects and is therefore, the preferable structure form to use for the QMF bank implementation (see Chapter 3).

4.4.1 Objective function performance landscape

An initial GA search for good sub-optimal 'seed' parameter values of the transformation function is essential for obtaining good results. It is well recognised that a good choice of GA parameters in this context is essential for efficient minimisation of the objective function. The outcome of the objective function to be minimised can vary significantly on the GA performance landscape. In some instances a bimodal response is observed in which case the choice of some GA parameters such as population size or mutation rate can become substantially relevant to the outcome of the converged GA and the final objective function result

[Oates et al, 1999]. An investigation of the transformation function parameters that would yield a minimum objective function value over an extensive GA performance landscape was conducted for a specific design example (example 3 in section 4.5). The objective function used in this example is shown in Equation 4.16 for which positive linear weighting is used. In an effort to achieve statistically significant results of the error function, each run of the GA over 100 generations was repeated 5 times and an average value taken. The population size was increased in steps of 10 and mutation rate ranged from 0 to 1. Figure 4.7 shows the results. Other GA parameters used in this example were reinsertion rate= 0.7 and generation gap= 0.9. On the basis of the minimum error function shown in Figure 4.7, the GA parameters selected for optimisation of the transformation function coefficients are; mutation rate = 0.2, population size = 200 and number of generations = 100. Other parameters used are same as mentioned above.

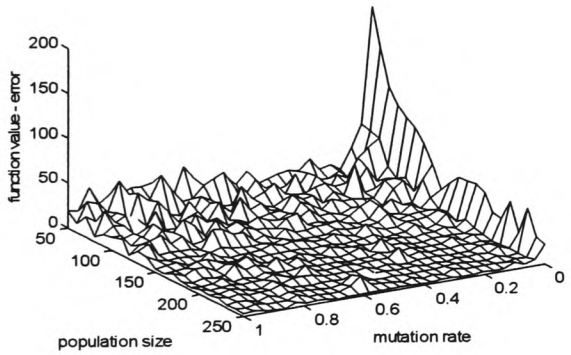

Figure 4.7 GA performance landscape

4.4.2 Optimisation using the GA 'creep' code

In order to conduct an efficient optimisation of the GA code and to draw a comparison with the standard gradient and non-gradient based methods, a special GA 'creep' code was developed. This code is incorporated in the search algorithm and becomes operative only after the first 20 generations of the GA code have been executed. The 'creep' code is a variation of the G-bit operator [Goldberg, 1983]. The G-bit improvement includes the following steps

- Select one or more of the best strings from the current population.
- Sweep bit by bit, performing successive one-bit changes to the subject string or strings, retaining the better.
- At the end of the sweep insert, the best structure (or k-best structures) into the population and continue the normal genetic search.

The above operator is used in binary encoding, however, the GA optimisation in the design examples considered in this work is based on the use of real valued strings. The variation of the G-bit operator that has been used in the development of the GA 'creep' code is

- Select one or more of the best strings from the current population.
- Modify the string or strings slightly, by adding a small random number to each gene, retaining the better.
- Repeat a number of times
- Insert the modified strings into the current population.

This 'creep' code has the effect of conducting a tumbling downhill Simplex type optimisation found to be useful for this application thus generating good quasi-optimal results. The algorithm for the 'creep' code is

:

if gen. > 20 then switch over to creep code

locate the best individual in the population

add a small random value to the parameters

repeat until 10 new individuals are formed

replace with 10 worst individuals from original

:

A typical generational display of the objective function values for design example 3 (see section 4.5) is shown in Figures 4.8 and 4.9. It can be seen that the GA minimises the objective function value fairly rapidly up to the 20th generation after which the algorithm switches over to the 'creep' code for further minimisation. The convergence is fairly slow without switchover to the 'creep' code. An example of the GA code using the 'creep' option for optimisation of the design example 3 is shown in Appendix D1.1.

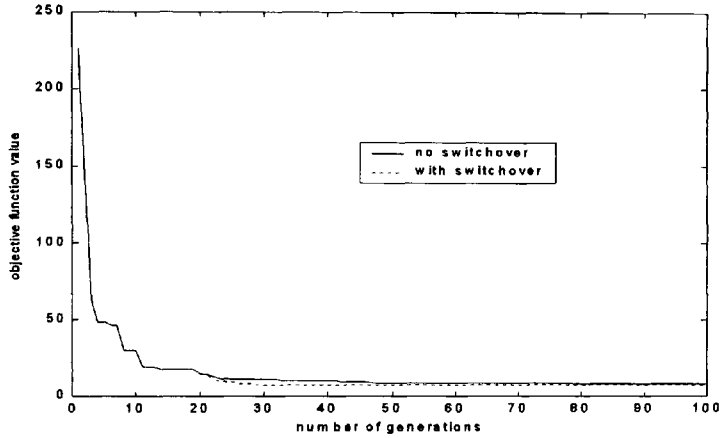

Figure 4.8 GA optimisation generational plot for design example 3 (see section 4.5)

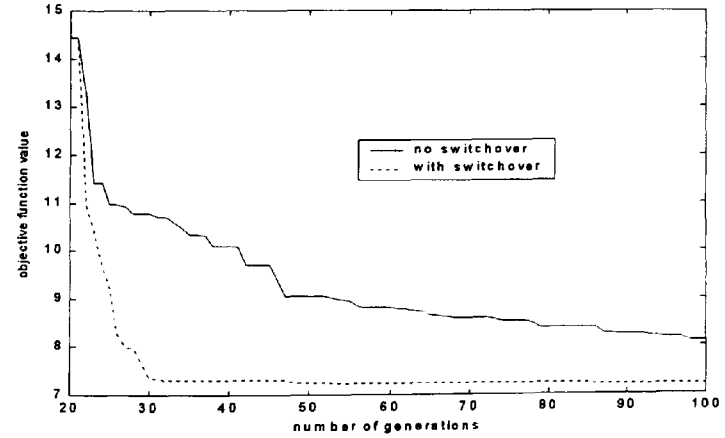

Figure 4.9 Expanded plot of Figure 4.8 starting from 20th generation.

4.4.3 Other standard optimisation methods used

Apart from the GA method, three other forms of standard optimisation procedures are used in this study for drawing a comparison. The first is the constrained optimisation technique based on the sequential quadratic programming methods that represent state-of-the-art in non-linear programming method and is executed using the constr.m function of the Matlab Optimisation toolbox. This method has the advantage of incorporating stability constraints required by the IIR filter design. However, it must be noted that the first stage GA searching for good 'seed' parameters of the transformation function also uses constrained search by awarding high penalty to individuals failing the stability test. This procedure assures the region for search is safe for generating stable solutions. This requirement becomes important with the subsequent two methods of unconstrained optimisation i.e. the gradient based Quasi-Newton method and the non-gradient based downhill Simplex method. The Quasi-Newton method of optimisation is implemented using the `fminu.m` function of the Matlab Optimisation toolbox. Note that the simplest gradient based methods such as the methods of steepest descent are highly inefficient for functions with long narrow valleys. However, the quasi-Newton gradient based method uses information about the slope of the function and formulates a quadratic model from the curvature information at each iteration to dictate the direction of search. This form of algorithm is the most favoured gradient based optimisation methods. The downhill Simplex method is a non-gradient based method and is implemented using the `fmins.m` function of Matlab Optimisation toolbox. This method is based on the procedure developed by Nelder and Mead [1965]. Such a search method uses only function evaluations and is generally most suitable for problems that are highly non-linear or have a number of discontinuities. Several design examples considered in this study are based on a hybrid approach i.e. first using GAs for searching over a wide landscape for promising valleys and then applying a standard optimisation method. The standard methods considered are; the gradient based SQP

`constr.m` function, the gradient based quasi-Newton method using the `fminu.m` function and the non-gradient based downhill Simplex using the `fmins.m` function of the Matlab Optimisation toolbox. Such a comparative study gives useful insight into the optimisation issues, the profile of the objective function landscape and the effectiveness of the optimisation process for the application considered in this study. See Appendix D1.2 for the Matlab m-file code.

4.5 Design examples and Results

Three design examples are considered in this section. These examples are of higher order transformations and correspond exactly to the design examples 3,4 and 5 considered by Tay [1998]. The design examples 1 and 2 of [Tay, 1998] result in trivial solutions and simple 'trial and error' procedure can be used for their optimisation. The exact correspondence of the three design examples considered here gives a good basis for comparative analysis and an inference of the efficiency and robustness of the optimisation process.

Design Example 1

Results in this example are compared with the results of example 3 of [Tay, 1998]. Transformation function (see Equation 4.14) with P_r=2, P_c=0 and b=0.5 is used and negative linear weighting is applied. The pass-band edge is $\omega_L = \pi/4$ and the frequency response roll-off value n=10 is used. Optimised parameter values and the corresponding objective function values are shown in Table 4.1.

Table 4.1 Comparative results for design example 1.

	c_1	c_2	d_1	d_2	Obj_fn
Ref.[Tay,1998]	-0.0989	0.6494	0.3637	-0.0526	**0.0441**
GA+ creep code	0.6497	-0.0991	0.3639	-0.0525	**0.0440**
GA+ `constr.m`	0.6500	-0.0992	0.3646	-0.0529	**0.0440**
GA+ Simplex	0.6500	-0.0992	0.3646	-0.0529	**0.0440**
GA+ Q-Newton	0.6500	-0.0992	0.3646	-0.0529	**0.0440**

The parameter values optimised for design example 1, as shown in Table 4.1, generate a trivial solution of the transformation function. The objective function of Equation 4.16 uses L=100 i.e. the summation is over 100 frequency points and is shown in the last column of Table 4.1. No significant improvement was achieved for this design example when compared with the optimised results of [Tay, 1998].

Design Example 2

Results in this example are compared with the results of example 4 of [Tay, 1998]. The transformation function is obtained using P_r=1, P_c=1 and b=0.5. The pass-band edge is $\omega_L = 3\pi/8$ and negative linear weighting is applied. The frequency response roll-off value used is n=10. Optimised parameter values and the corresponding objective function values are shown in Table 4.2. This design example is of a higher order transformation function involving six variables. Once again, the last column of Table 4.2 shows the results of the objective function calculated over 100 frequency points. A small improvement of the GA optimised and of the hybrid optimised (i.e. the GA optimisation followed by the standard optimisation methods) results is evident for this design example when compared with the results of Tay [1998].

Table 4.2 Comparative results for design example 2.

	c	d	r	ϕ	p	ψ	**Obj_fn**
Ref. [Tay, 1998]	0.7475	0.4978	0.1897	1.0304	0.1082	-0.9987	**0.1775**
GA+ creep code	0.7478	0.4965	0.1908	1.0315	0.1045	-0.9866	**0.1770**
GA+ Constr.m	0.7490	0.5000	0.1918	1.0318	0.1096	-1.0012	**0.1764**
GA+ Simplex	0.7491	0.5001	0.1918	1.0318	0.1097	1.0013	**0.1764**
GA+ Quasi-Newton	0.7491	0.5001	0.1918	1.0318	0.1097	1.0013	**0.1764**

Design Example 3

Results in this example are compared with the results of example 5 of Tay [1998]. The transformation function is obtained using P_r=1, P_c=1 and b=0.7. The passband edge is $\omega_L = 0.43\pi$ and positive linear weight is applied. The frequency response roll-off value of n=25 is used. Optimised parameter values and the corresponding objective function values are shown in Table 4.3. Once again, the results of the objective function show improvements of the GA and the hybrid optimised results when compared with the results of Tay [1998]. Of special interest here are the results of the GA+creep code that shows an improvement over the singular optimisation process of the 'constr.m' functions of Matlab Optimisation toolbox. However, further tests using the GA+gradient based quasi-Newton and the GA+non-gradient based downhill Simplex algorithms show further improvements of the results as seen in Table 4.3. In general, it is significant to mention that the GA and the hybrid optimised results were derived in a direct manner without the need for trial 'seed' values. The results obtained were consistently good thus giving confidence to the design engineer for a likely optimal outcome. Furthermore, this process can be applied to higher order transformation functions with larger number of parameter values thus giving greater flexibility for a near optimal design implementation.

Table 4.3 Comparative results for design example 3

	c	d	r	ϕ	p	ψ	**Obj_fn**
Ref.[8] Constr.m	0.7740	0.6704	0.2872	1.1058	0.2398	-1.1156	**7.3635**
GA+ creep code	0.7784	0.6761	-0.2930	2.0348	0.2446	-1.1164	**7.1902**
GA+ Constr.m	0.7785	0.6764	0.2938	-1.1086	0.2461	-1.1191	**7.1875**
GA+ Simplex	0.7785	0.6765	0.2938	-1.1086	0.2461	-1.1191	**7.1875**
GA+ Quasi-Newton	0.7785	0.6764	0.2938	-1.1086	0.2461	-1.1191	**7.1875**

4.5.1 Some remarks on the optimisation process

Although, in general, real valued GAs are not highly efficient in optimisation of engineering design problems, the hybrid procedure used in this chapter has generated good results for the application considered. A number of trial GA runs showed that for most of the time, the GA generated near-optimal minima value although the initial population was randomly generated. This method is thus more robust and effective when compared to the singular gradient-based method of constrained optimisation used in [Tay, 1998] which is highly sensitive to starting point parameter values. It must also be recognised that for higher order transformations generating sharper roll-off filters, the number of transformation parameters increases significantly. A GA based optimisation procedure searching for promising optima valleys becomes even more relevant in such situations. This work clearly demonstrates the importance of hybrid GAs for optimising fitness landscapes that are dominated by local minima on a large scale but are smooth and well behaved in a local area. Clearly, methods of categorising landscapes in order to identify the most effective hybrid approach is an important issue that has yet to be fully investigated.

4.6 Design of Filters for the QMF bank

In the previous section (section 4.5), a hybrid optimisation process was considered for optimising the parameters of the transformation function Z. The prototype filters can then be generated through Equation 4.11. In this section, the low-pass/high-pass analysis/synthesis filters of Equations 4.7 and 4.8 will be derived from the optimised prototype filters. The design example 2 of section 4.5 is considered here. The optimised parameter values obtained using the hybrid GA are taken from Table 4.2 and listed here in Table 4.4. The choice of these values gives a minimal objective function value.

Table 4.4 Optimised parameter values for design example 2

c	d	r	ϕ	p	ψ	Obj_fn
0.7491	0.5001	0.1918	1.0318	0.1097	1.0013	0.1764

4.6.1 Design and Simulation in Matlab

A Matlab based code was developed for deriving the four filters H_0, F_0, H_1 and F_1 based on the values of the parameters in Table 4.4 (see Appendix D2.1). Table 4.5 shows the coefficient values of the four filters. Note that H_0b and H_0a represent the numerator and denominator coefficient values of the low pass analysis filter H_0 respectively. Similarly H_1 represents the high pass analysis filter and F_0 and F_1 are low pass and high pass synthesis filters respectively. Figures 4.10(a) and 4.10(b) show the magnitude response of the analysis and synthesis filters respectively. The phase responses of the four filters are shown in Figure 4.11. It can be seen that the phase response is linear in the pass band region of the filters. This property contributes to the requirement for perfect reconstruction of the output signal.

Table 4.5 Coefficient values of the QMF bank filters

H_0b	H_0a	F_0b	F_0a	H_1b	H_1a	F_1b	F_1a
-0.0002	1.0000	-0.0000	1.0000	0.0000	1.0000	-0.0002	1.0000
0	0	0	0	0	0	0	0
0.0013	0.7636	0.0000	1.1454	-0.0000	1.1454	0.0013	0.7636
0	0	0	0	0	0	0	0
-0.0089	0.0515	-0.0001	0.2959	0.0001	0.2959	-0.0089	0.0515
0.0126	0	-0.0001	0	-0.0001	0	-0.0126	0
0.0140	-0.0240	0.0006	-0.0343	-0.0006	-0.0343	0.0140	-0.0240
-0.0457	0	0.0008	0	0.0008	0	0.0457	0
-0.0174	0.0068	-0.0017	-0.0002	0.0017	-0.0002	-0.0174	0.0068
0.2321	0	-0.0056	0	-0.0056	0	-0.2321	0
0.5200	-0.0006	0.0130	0.0035	-0.0130	0.0035	0.5200	-0.0006
0.5550	0	0.0070	0	0.0070	0	-0.5550	0
0.3645	0.0000	-0.0404	-0.0006	0.0404	-0.0006	0.3645	0.0000
0.1620	0	-0.0078	0	-0.0078	0	-0.1620	0
0.0386	0	0.2128	0.0001	-0.2128	0.0001	0.0386	0
-0.0200	0	0.5085	0	0.5085	0	0.0200	0
-0.0180	0	0.6485	-0.0000	-0.6485	-0.0000	-0.0180	0
0.0027	0	0.5669	0	0.5669	0	-0.0027	0
0.0051	0	0.3598	0.0000	-0.3598	0.0000	0.0051	0
0	0	0.1514	0	0.1514	0	0	0
-0.0004	0	0.0200	0	-0.0200	0	-0.0004	0
0	0	-0.0172	0	-0.0172	0	0	0
0.0000	0	-0.0107	0	0.0107	0	0.0000	0
		-0.0009	0	-0.0009	0		
		0.0035	0	-0.0035	0		
		0.0023	0	0.0023	0		
		-0.0003	0	0.0003	0		
		-0.0004	0	-0.0004	0		
		0.0000	0	-0.0000	0		
		0.0001	0	0.0001	0		
		0	0	0	0		
		-0.0000	0	-0.0000	0		
		0	0	0	0		
		0.0000	0	0.0000	0		

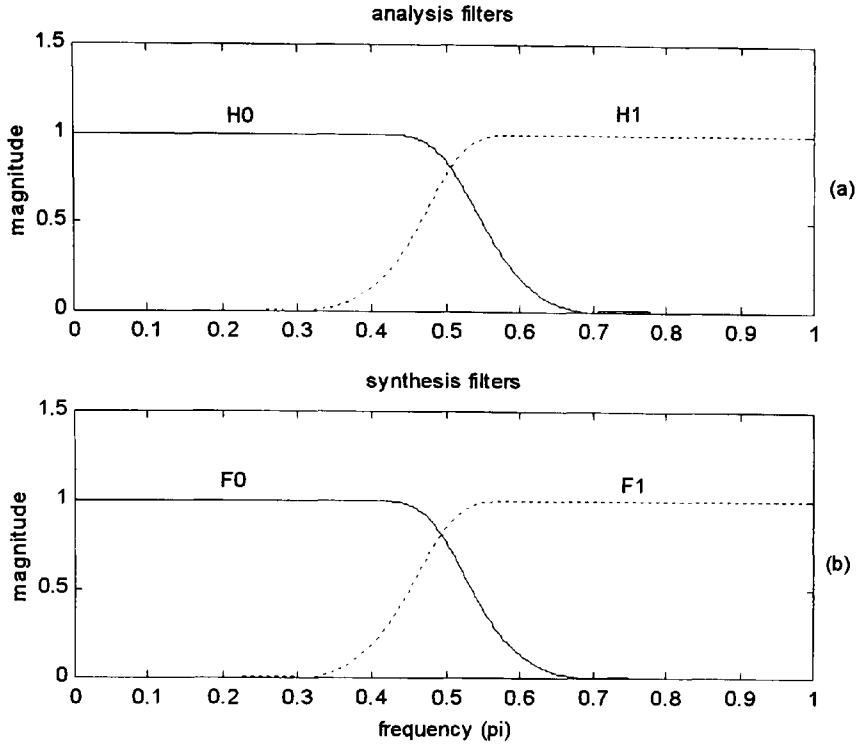

Figure 4.10 Magnitude response of the analysis filters (a) and synthesis filters (b).

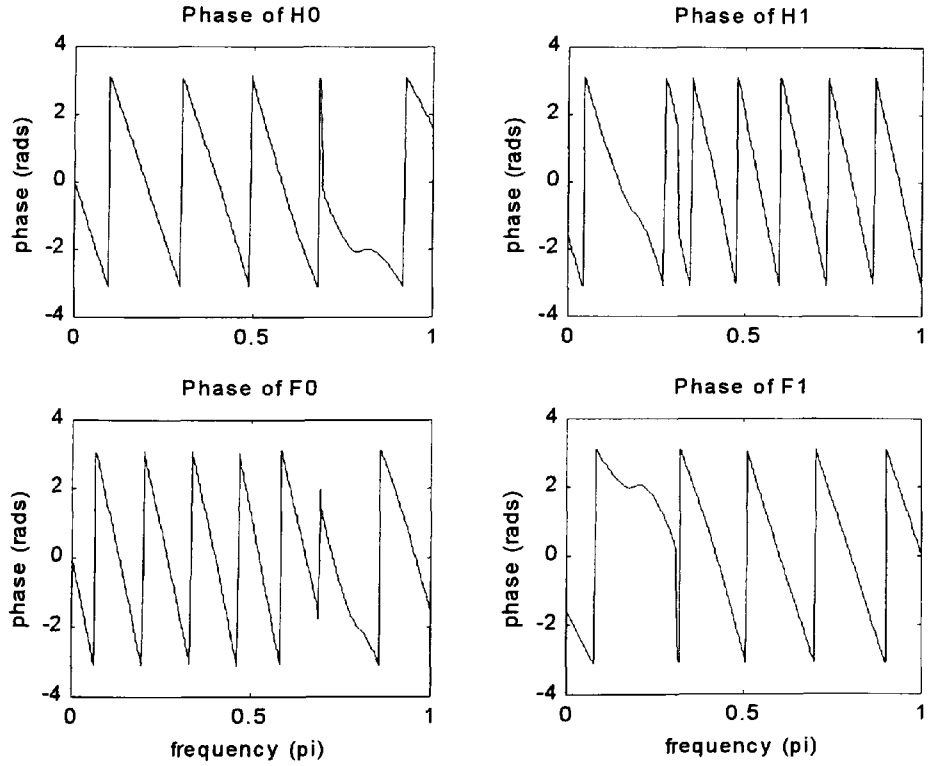

Figure 4.11 Phase response of the QMF filters.

In order to determine the amplitude distortion of the QMF bank, the overall transfer function of the QMF bank was derived using the transfer function of individual IIR filters. The amplitude distortion and group delay for the QMF bank are shown in Figure 4.12. These results clearly indicate close proximity to perfect reconstruction of the output signal. Further tests were conducted using the Simulink toolbox of Matlab. The QMF bank implementation in Simulink is shown in Figure 4.13. The input signal used is a random signal with uniform distribution in the range 1 to −1. The error signal is then obtained from the difference between the output signal and a delayed version of the input signal. The error signal for the optimised design example 2 is shown in Figure 4.14. Note that this error signal has a maximum error magnitude of approximately 1.5×10^{-15} that is close to the limits of the software mathematical error bounds. The Simulink test results demonstrate the perfect reconstruction property of the QMF bank based on IIR filters designed using the optimised coefficients of the transformation of variable technique.

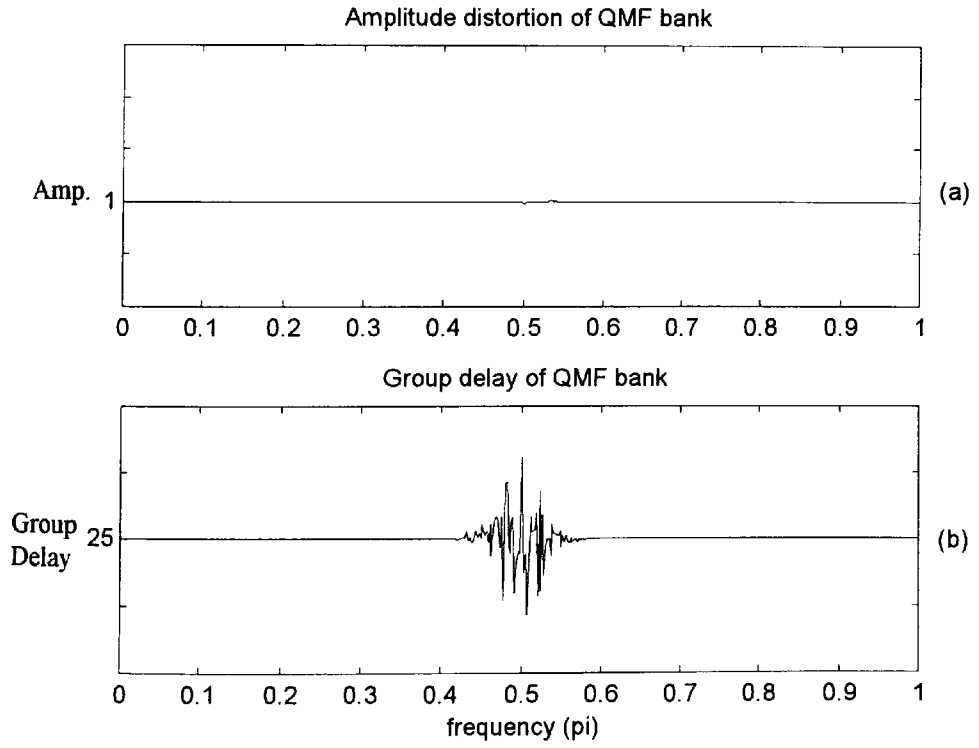

Figure 4.12 Amplitude distortion (a) and group delay (b) of the overall transfer function.

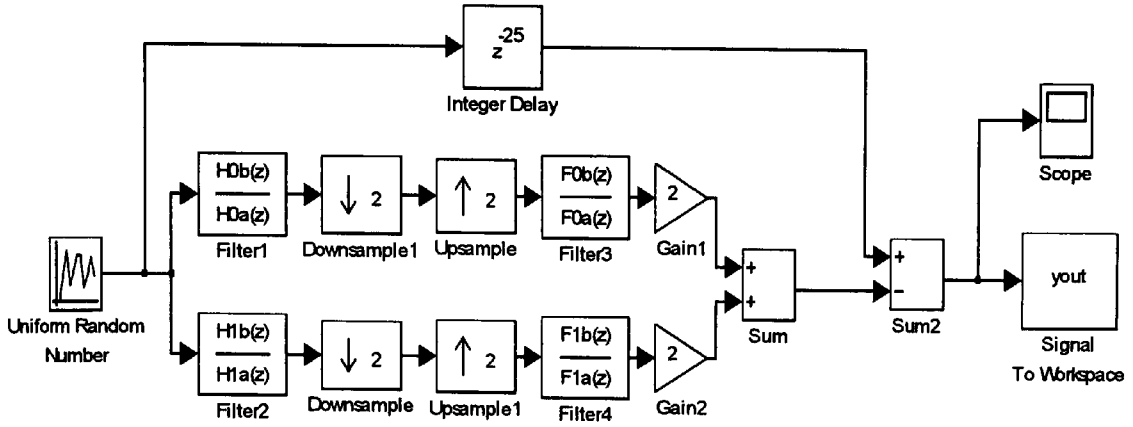

Figure 4.13 Simulink model of the QMF bank

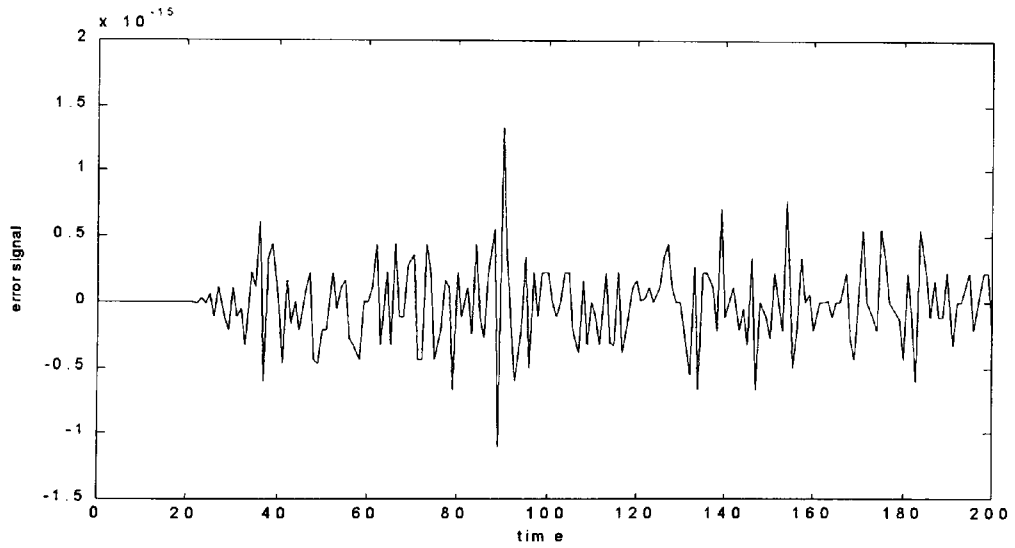

Figure 4.14 Error signal of the QMF bank for a random input signal (design example 2).

4.6.2 Finite word length constraints for real-time implementation

The filter coefficients shown in Table 4.5 are for direct form IIR filter implementation. These are infinite precision values generated using Matlab. However, for real-time implementation, there are two further issues that must be considered. These are; firstly, finite word length

constraints due to either hardware limitations or efficient throughput requirements and secondly, using the IIR filter structure of a second order cascade form. It has been seen in Chapter 3 that second order cascade structures are less sensitive to FWL constraints and is thus a preferred option. The coefficient values of the four filters in Table 4.5 must, therefore, be converted to second order cascade form as given by Equation 3.5. The Matlab function that is used for this conversion is `sos=tf2sos(b,a)`.

The second-stage GA optimisation for finite word length constraint of the filter coefficient values (see Figure 4.6) was conducted for the low pass filter H_0, using 5 bits to represent the coefficient values. The simply rounded values and the GA optimised values for the design examples 2 and 3 are shown in Tables 4.6 and Table 4.7 respectively. The magnitude response of the low pass H_0 filter for the design examples 2 and 3 are shown in Figures 4.15 and 4.16 respectively. The GA parameters used in these examples are; *number of individuals*= 100, *number of generations*= 20, *reinsertion rate*= 1.0 and *generation gap*= 0.8. The GA optimised response clearly indicates improvements for smaller number of bits that can be used as implementation for fast processing on dedicated high-speed low-bit hardware for real-time realisation.

Table 4.6 H_0 coefficient values using 5 bits for design example 2.

```
xround = [br, ar]
br = 0  0  0  0  0  0  0  -1  0  3  8  8  5  2  1  0  0  0  0  0  0  0  0
ar = 15 0  11 0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0

x_gaopt=[bo, ao]
bo = 0  0  0  0  0  0  0  -1  0  4  8  8  5  2  1  0  0  0  0  0  0  0  0
ao = 15 0  11 0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
```

Table 4.7 H_0 coefficient values using 5 bits for design example 3.

```
xround = [br, ar]
br = 0  0  0  0  0  0  0 -1  0  3  8  9  7  3  0  0  0  0  0  0  0  0  0
ao =15  0 14  0  1  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0

x_gaopt=[bo, ao]
bo =0  0  0  0  0  0  0  0 -1  0  3  8  9  7  3  1  0  0  0  0  0  0  0  0
ao =15 0 13  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
```

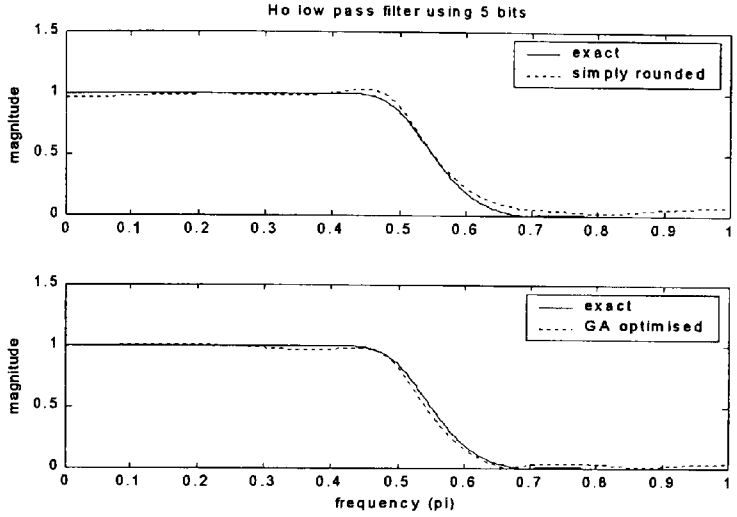

Figure 4.15 Magnitude response of H_0 low pass filter for design example 2.

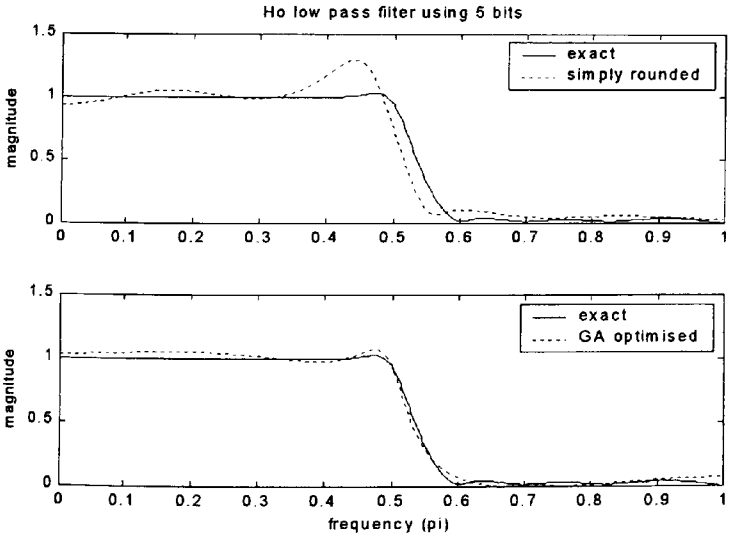

Figure 4.16 Magnitude response of H_0 low pass filter for design example 3.

4.7 Real time implementation issues

As mentioned in section 4.6, the four direct form IIR filters of the QMF bank must firstly be converted to a second order cascade form of Figure 4.17 using the Matlab file sos=tf2sos(b,a). For real time implementation on a target DSP system the coefficients required to be stored from low to high data memory locations for 'n' second order sections must be in the form $-a_{21}$, $-a_{11}$, b_{21}, b_{11}, b_{01}, ..., $-a_{2n}$,$-a_{1n}$, b_{2n}, b_{1n}, b_{0n}. Note that the coefficients a_{0j} are always 1 so there is no need to store these coefficients. The delay elements are similarly stored from low to high data memory in the form $d_n(m)$, $d_n(m-1)$, $d_n(m-2)$, ...,$d_1(m)$, $d_1(m-1)$, $d_1(m-2)$. A typical filtering algorithm using the Texas Instrument's TMS320C50 DSP assembly code form is shown in Figure 4.18 [Texas Instruments, 1991].

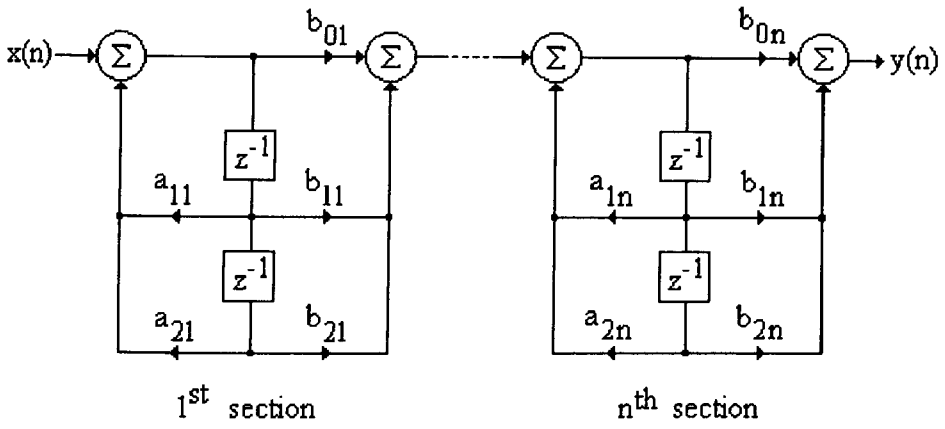

Figure 4.17 Cascade form structure of IIR digital filter

```
; AR1 → d1(m-2)
; AR2 → -a21
; AR3 → input sample (Q15 number)
; DP=0, PM=0, ARP=3
ZPR
LACC *,15,AR1
SPLK #2,INDX
SPLK #N-1,BRCR

RPTB ELOOP-1
LOOP:
        LT      *-,AR2
        MPYA    *+,AR1
        LTA     *-,AR2
        MPY     *+
        LTA     *+,AR1
        SACH    *0+,1
        MPY     *-
        LACL    #0
        LTD     *-,AR2
        MPY     *+,AR1
        LTD     *-,AR2
        MPY     *+,AR1
ELOOP:
        APAC
        SACH OUTPUT,1
```

Figure 4.18 An example of a typical filter algorithm using the TMS320C50 assembly code.

A Matlab file sos2v.m (Appendix D2.2) generates a vector of coefficients in the ordered form for use with the filtering algorithm given a second order section matrix. The resulting vector can be converted into a hexadecimal Q-15 format and then imported into the TMS320C50 assembly code [Texas Instruments, 1991]. The Matlab file sos2v.m also returns a vector E containing the rows of the second order section matrix having the 'b' (numerator) coefficients that lie outside the range $-1<coeff<0.999965$. Such coefficient values cannot be represented in the Q-15 format so the numerator coefficients of these sections must be scaled down by some relevant factor before conversion to the Q-15 format. The Matlab file scsos.m (Appendix D2.2) conducts a scaling procedure thereby returning the scaled second order section filter coefficients.

4.7.1 Testing the filters using the TMS302C50 simulator

The TMS320C50 simulator was used to test the transfer functions of the analysis and synthesis filters [Texas Instruments, 1994]. The simulator is a software programme that mimics the functions of the fixed point TMS320C50 digital signal processor. This process is useful for debugging the assembly code before down loading on to the real-time target system in the common object file format (COFF) form. The frequency response of the individual filters can be obtained by applying an impulse to the input of the simulator and then evaluating the Fast Fourier Transform (FFT) of the impulse response in Matlab [Baicher and Sherrington, 1996]. Figures 4.19 and 4.20 show the magnitude and phase responses of the four filters respectively. Note that the 16 bit FWL constraint on the coefficient values has not affected the frequency response of the filters and is identical to the frequency responses obtained previously (Figures 4.10 and 4.11).

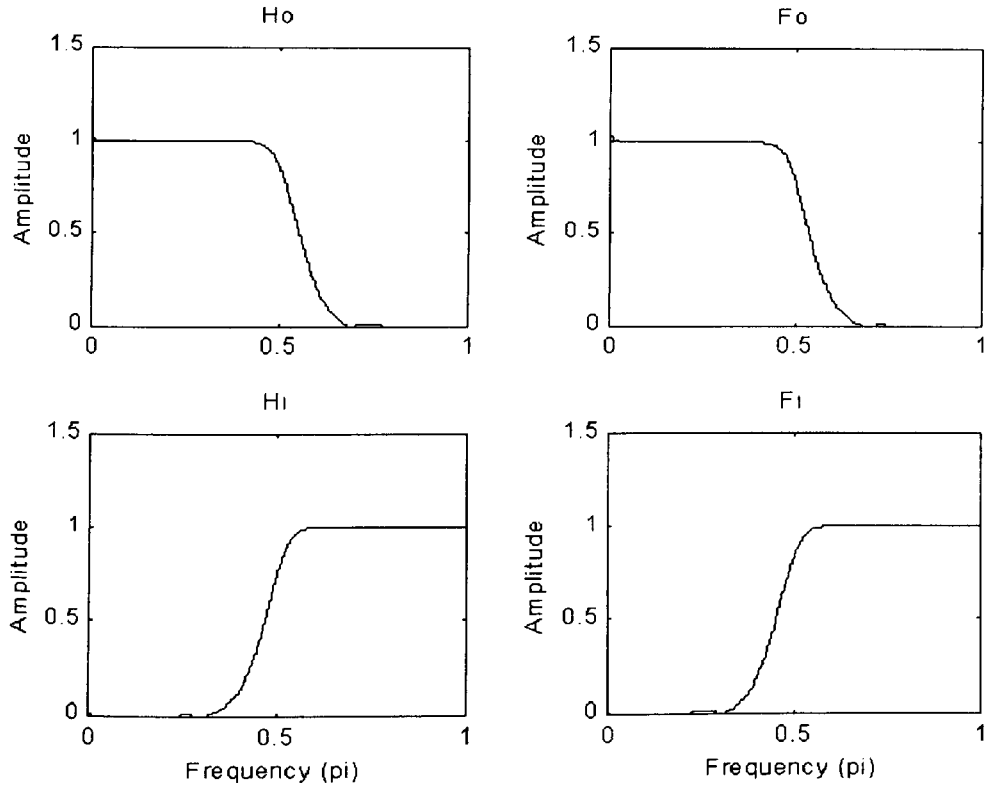

Figure 4.19 Magnitude response of the four QMF filters.

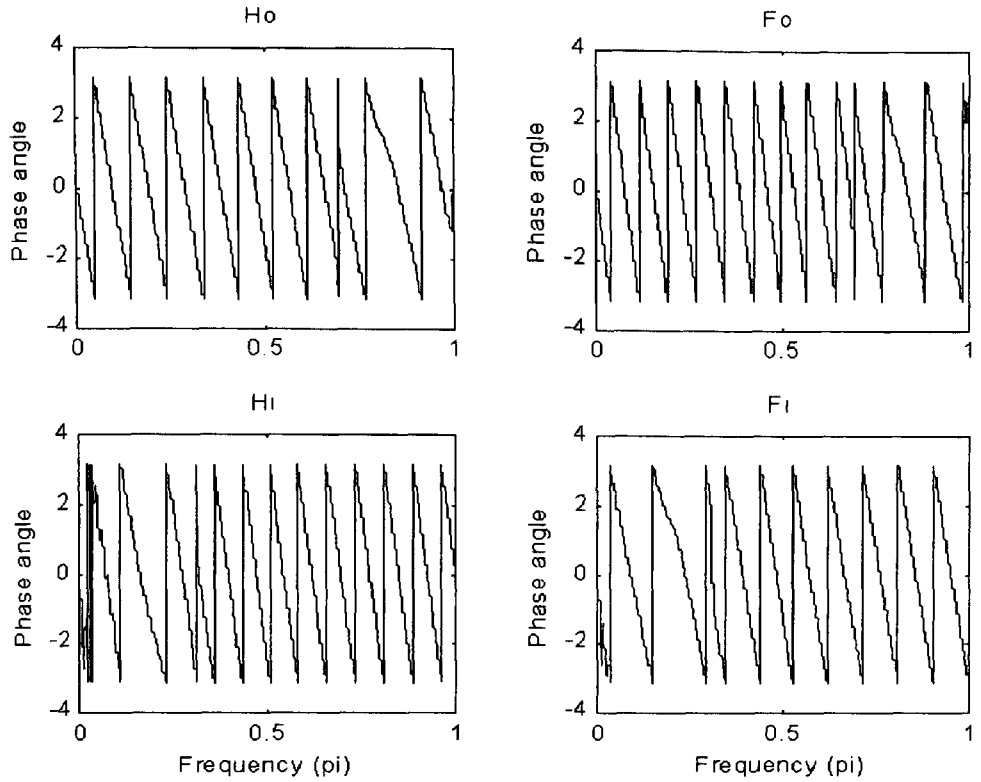

Figure 4.20 Phase response of the four QMF filters.

4.7.2 Polyphase decomposition – a computationally efficient realisation

It is well established that polyphase decomposition of digital filters in realisations of multirate filter operations can facilitate computational savings in software and hardware implementations [Mitra, 1998], [Vaidyanathan, 1993]. Such polyphase implementations are useful for linear phase FIR filters and in certain structural forms of IIR filters. In order to derive the 2-branch polyphase decomposition of an IIR filter having a transfer function H(z) = N(z) / D(z), it is necessary to express the filters in the form $N'(z) / D'(z^2)$. The form of transformation function of Equation 4.14 ensures that the denominator of M(z) and hence the denominators of $H_T(M(z))$ and $F_T(M(z))$ are functions of z^2. This property of the denominator polynomial is clearly evident by examining the denominator coefficients of the filters shown in Table 4.5. Note that every alternative coefficient of the denominator is a zero. Therefore, the transfer function H(z) of the resulting filters will be of the form

$$H(z) = \frac{N(z)}{D(z^2)} = \frac{h_0 + h_1 z^{-1} + h_2 z^{-2} + \ldots + h_n z^{-n}}{D(z^2)} \qquad 4.24$$

Separating the even and odd-indexed terms of the numerator, the transfer function is of the form

$$H(z) = \frac{h_0 + h_2 z^{-2} + \ldots + h_m z^{-m}}{D(z^2)} + z^{-1} \frac{h_1 + h_3 z^{-2} + \ldots + h_k z^{-(k-1)}}{D(z^2)} \qquad 4.25$$

where $m = n$ and $k = n-1$ for n = even

or $m = n-1$ and $k = n$ for n = odd.

The polyphase components of H(z) are

$$E_0(z) = \frac{h_0 + h_2 z^{-1} + \ldots + h_m z^{-m/2}}{D(z)} \qquad 4.26$$

and

$$E_1(z) = \frac{h_1 + h_3 z^{-1} + \ldots + h_k z^{-(k-1)/2}}{D(z)} \qquad 4.27$$

The polyphase components are easily derived from the filter coefficients given in Table 4.5. The numerator values of the polyphase components are given by taking alternative coefficients of the numerator values and the denominator values are obtained by starting with the first coefficient for the denominator polynomial and taking every other value (i.e. ignoring the zero value coefficients). By expressing the analysis and synthesis filters of the QMF bank of Figure 4.2 in polyphase form as shown in Figure 4.21, the efficiency of the system can be improved without changing its characteristics.

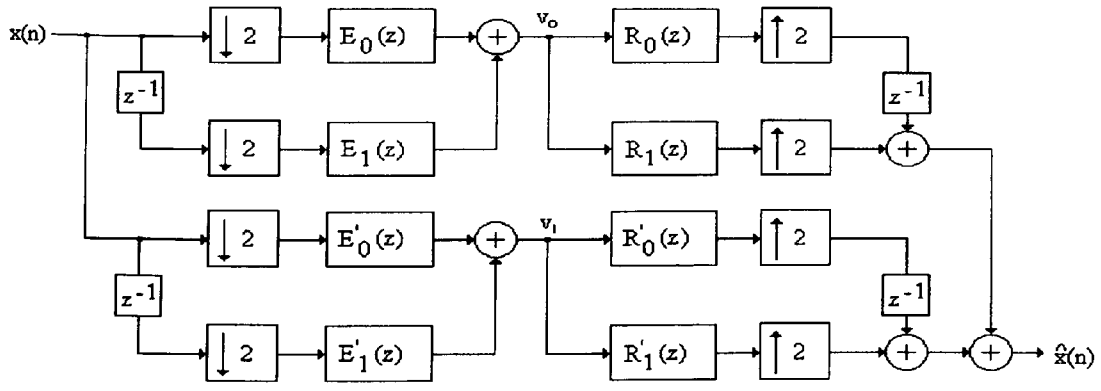

Figure 4.21 QMF bank in polyphase form.

The system of Figure 4.21 was tested using the Simulink toolbox of Matlab as shown in Figure 4.22. The results obtained are identical to the results of section 4.6.1 as shown in Figure 4.14. The system of Figure 4.21 is more efficient since the processor speed of the filters is effectively halved without affecting the output. The process of polyphase decomposition achieves this.

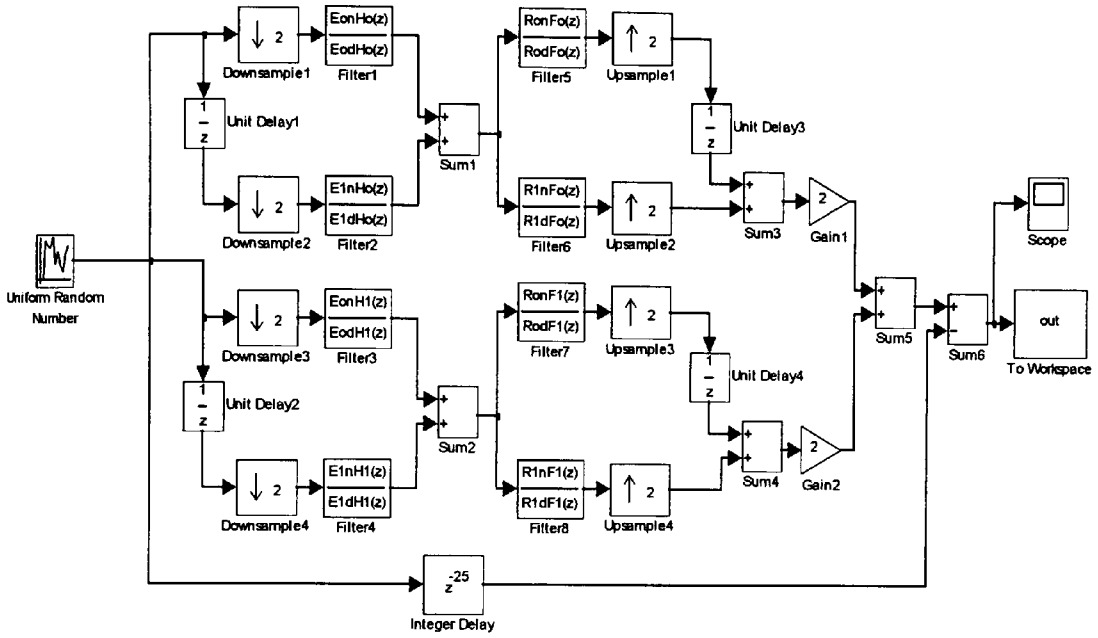

Figure 4.22 Simulink model of the polyphase form QMF bank.

4.7.3 An 8 bit QMF bank in polyphase form

The block diagram of Figure 4.21 can be implemented using 8 bit fixed point arithmetic and has potential application in pulse code modulation (PCM) telephony signals [Crochiere, 1981]. The lower and higher frequency sub-bands of the QMF bank can be encoded with 8 and 4 bits respectively to provide a data rate compression of 4/3. Figure 4.23 shows how the scaled polyphase components are affected by 8-bit coefficient representation, when cascade realisation is used. The solid lines show the desired magnitude response and the dashed lines show the rounded response.

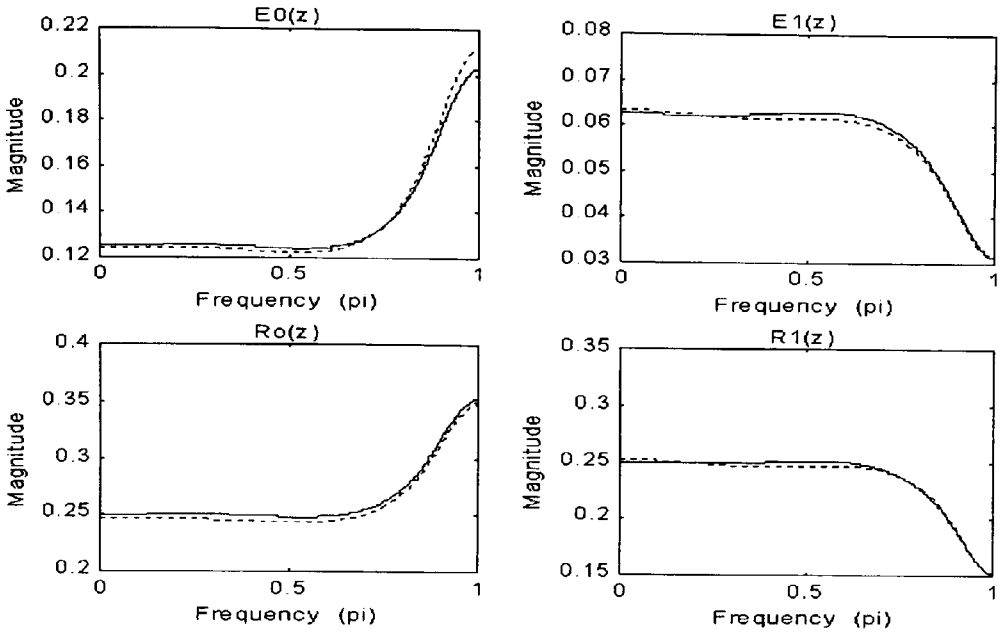

Figure 4.23 Effect of 8-bit coefficient wordlength on polyphase components. Solid line shows desired response and dashed line is for rounded coefficient response.

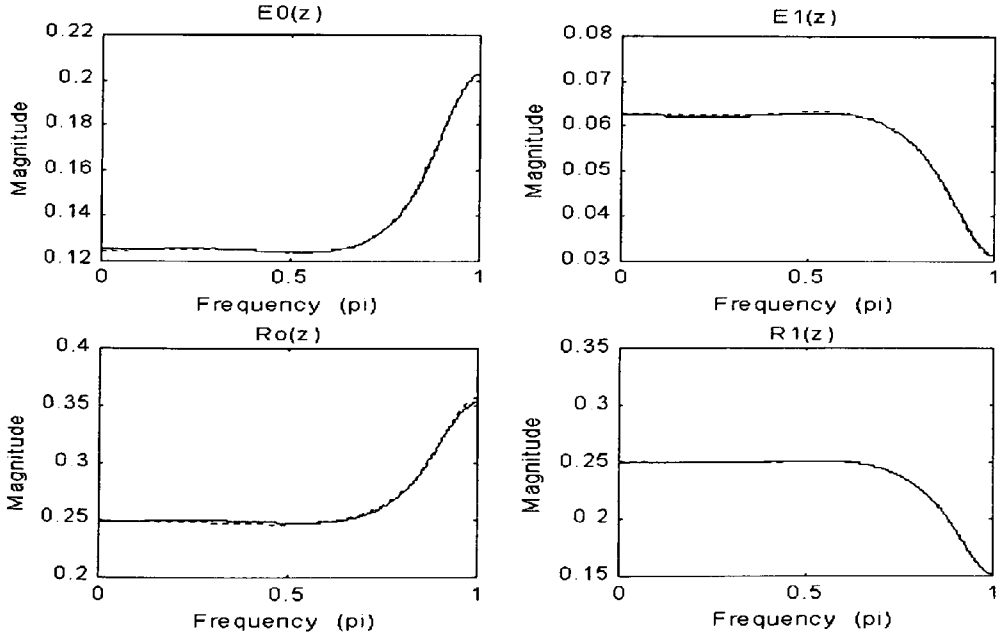

Figure 4.24 GA optimised 8 bit coefficient polyphase components. Solid line shows desired response and dashed line is for GA optimised coefficient response.

Polyphase components E_0 and E_1 refer to the filter H_0 and R_0 and R_1 refer to the filter F_0. The effects of 8 bit rounding appear small, with polyphase component E_0 having the largest distortion. This is a relatively simple GA optimisation problem. Figure 4.24 shows the GA optimised results using the sos_ga.m file (see chapter 3). The GA optimised (dashed line) and the high precision actual responses (solid line) are almost identical. The phase response was not affected. The polyphase components of H_1 and F_1 are also optimised to produce similar results.

4.7.4 Sub-band coding of speech signals

It is well recognised that higher frequency band speech signals have lower energy levels and the lower frequency band signals have higher energy levels [Bellamy, 2000]. This characteristic of the speech signal can be exploited to improve the coding gain of the digital telephony signals. The bit rate assigned to each sub-band can be optimised to match the hearing perception of the human ear. In particular, larger number of bits per sample can be assigned to the lower frequency band where it is important to preserve the pitch and structure of human voice sounds. However, fewer numbers of bits per sample can be used for higher frequency band where noise-like segmented sounds have less effect on the reproduction quality. In general, a number of sub-bands can be used to optimise the coding compression. The scheme due to Crochiere [1981] splits the signal into four equal bands of 0 to 1kHz, 1 to 2kHz, 2 to 3kHz and 3 to 4kHz. The first band is further split in to two bands of 0 to 0.5kHz and 0.5 to 1kHz. Thus there is a pruned tree effect of five bands. However, for simplicity and ease of real-time implementation, a structure of splitting the speech signal into two equal bands using the QMF bank optimised and developed in this chapter will be considered here.

4.8 Real time implementation on a TMS320C50 DSK

The real time hardware used to test and run the programs in this work is the TMS320C50 DSP Starter Kit (DSK) [Texas Instruments, 1996]. Some of the main features of the DSK are

- 40 MHz clocking
- On-chip 10k RAM
- Single access memory can be configured as program memory or data memory
- 14 bit A/D and D/A conversion using the TLC32040 Analogue Interface Circuitry
- Standard RCA connectors for analogue input/output
- PC communication through RS232 port
- On-board EPROM controls the communication
- Expandability through expansion connectors.

The DSP code can be downloaded on to the DSK and run through the RS232 serial port using the DSK Debugger program or the DSK Loader programs. The DSK Debugger offers debugging features similar to the simulator features (section 4.7.1). Also provided is the DSK Assembler. This generates executable *.dsk files that can be loaded and run on the DSK. The common object file format (COFF) files of the type *.out can also be loaded and run on the DSK. Figure 4.25 shows the basic block diagram of the TMS320C50 DSK. The 32K PROM contains the kernel program for boot loading and RS232 communication control.

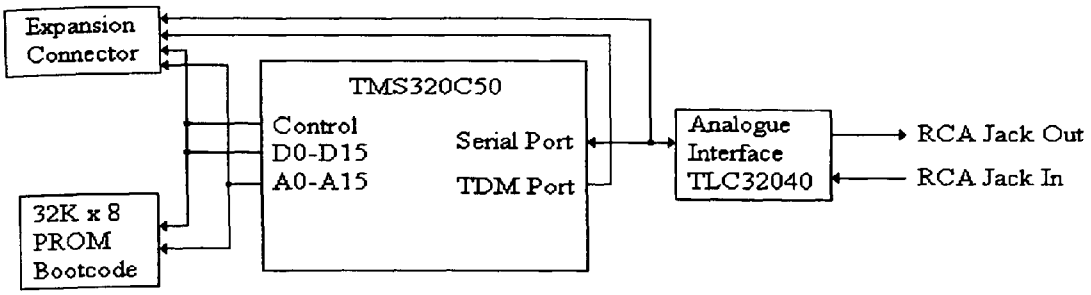

Figure 4.25 A basic block diagram of the TMS320C50 DSP kit.

4.8.1 Companding in a QMF bank structure

In this section, an appropriate method of implementing the companding process in a quadrature mirror filter bank will be considered. The compression procedure will normally distort the spectral properties of speech and therefore it cannot be applied at a point preceding the analysis filter bank. A more appropriate approach would be to compress each sub-band signal according to its characteristics in order to ensure that the useful properties of each sub-band are not altered. This is shown in Figure 4.26. It is assumed that the lower frequency segment of the speech has a peak magnitude of 1 and the higher frequency segment has a peak magnitude of 0.25. If each sub-band is compressed according to its peak magnitude then an improvement in signal to quantisation noise ratio (SQNR) is achievable while still being able to compress the data rate by encoding $v_1(n)$ with fewer bits. Signals $v_0(n)$ and $v_1(n)$ must be expanded by a matching expander before being processed by the synthesis filters.

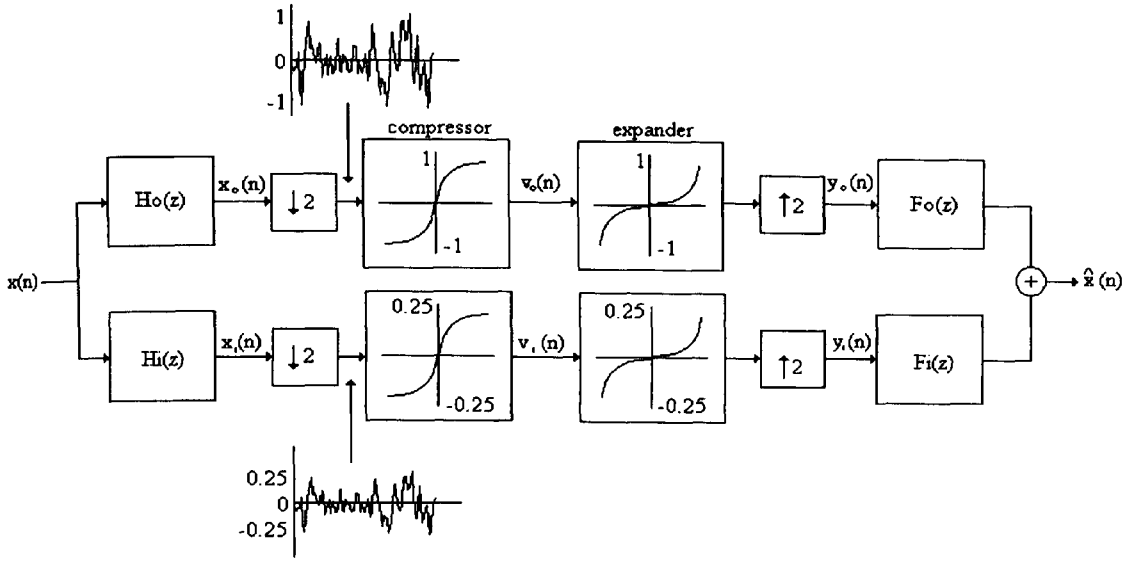

Figure 4.26 Sub-band compression and expansion of a QMF bank.

4.8.2 Testing and comparison of several QMF banks

A set of four two-channel QMF banks employing uniform encoding i.e. without companding and four two-channel QMF banks using the non-linear A-law companding (with A=87.56) were developed and tested on a TMS320C50 DSP kit. Look-up tables provide a mapping between the 12-bit codes and the compressed 8-bit codes [Bellamy, 2000]. Table 4.8 shows the modified compression and expansion mappings. Bits indicated by 'X' are lost during the process of compression-expansion. Figure 4.27 shows the structure of the 8-bit compressed code. Both sets use 8-8, 8-5, 8-4 and 7-5 encoding (x-y encoding stands for x bits for the lower frequency channel and y bits for the higher frequency channel). The encoding schemes are based on channel signal level and significance. The data compression achievable for the 8-8, 8-5, 8-4 and 7-5 encoding are 64 kbps, 56 kbps, 48 kbps and 48 kbps respectively, assuming sampling rate of 8 kHz is used. Companding is performed using Table 4.8 for the lower frequency channel. A similar table for the higher frequency channel is used based on 8 segments with 2 bits of significance for the 5 bit encoding (compression from 6 bits) and 8 segments with 1 bit of

significance of the 4 bit encoding (compression from 5 bits). The first set of QMF banks (i.e. no companding) uses 8-bit A/D and D/A conversion while the second set uses 12-bit A/D and D/A conversion.

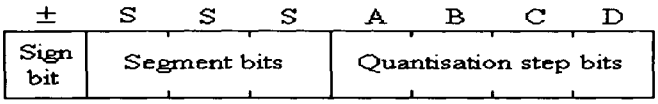

Figure 4.27 Structure of the 8-bit compressed code

Table 4.8 Modified compression and expansion mappings.

12-bit linear code	8-bit compressed code	12-bit expanded code
00000000**ABCD**	0000**ABCD**	00000000**ABCD**
00000001**ABCD**	0001**ABCD**	00000001**ABCD**
0000001**ABCDX**	0010**ABCD**	0000001**ABCD**1
000001**ABCDXX**	0011**ABCD**	000001**ABCD**10
00001**ABCDXXX**	0100**ABCD**	00001**ABCD**100
0001**ABCDXXXX**	0101**ABCD**	0001**ABCD**1000
001**ABCDXXXXX**	0110**ABCD**	001**ABCD**10000
01**ABCDXXXXXX**	0111**ABCD**	01**ABCD**10000
11111111**ABCD**	1111**ABCD**	11111111**ABCD**
11111110**ABCD**	1110**ABCD**	11111110**ABCD**
1111110**ABCDX**	1101**ABCD**	1111110**ABCD**1
111110**ABCDXX**	1100**ABCD**	111110**ABCD**10
11110**ABCDXXX**	1011**ABCD**	11110**ABCD**100
1110**ABCDXXXX**	1010**ABCD**	1110**ABCD**1000
110**ABCDXXXXX**	1001**ABCD**	110**ABCD**10000
10**ABCDXXXXXX**	1000**ABCD**	10**ABCD**100000

The various QMF bank models were compared based on the Mean Opinion Score (MOS) test [Vaidyanathan, 1993], [Porat, 1997]. This is a commonly used measure of speech quality and is based on evaluating the average speech quality on a scale of 1 to 5 where 1 is bad and 5 is excellent. A sample of 10 individuals was taken to conduct the tests. The individual scores for each encoding scheme were averaged to give the MOS. Table 4.9 shows the results of the test including the MOS values calculated. The results clearly show that the companding process yielded higher MOS values when compared to simple encoding of sub-band signals for identical encoding schemes. It must be emphasised that although the sample of ten individuals used in this test is relatively small and thus statistically not significant, however, a trend has clearly emerged that deserves further extensive investigation in the future.

Table 4.9 Mean opinion score (MOS) results for ten individuals. LP=low pass, HP=high pass.

Companding	LP	HP	5	4.5	4	3.5	3	2.5	2	1.5	1	MOS
No	8	8			••••	•••	•••					3.6
No	8	5					•••••	•••	•••			2.8
No	8	4					•••	••	••	••	•	2.2
No	7	5						•••	••••	••	•	2
Yes	8	8	•	••••	••	•••						4.2
Yes	8	5			••••	••	••••					3.5
Yes	8	4			•••	•••	••	••				3.4
Yes	7	5	•	••••	••••	•						4.3

4.9 Discussion of results and contributions of this chapter

A new design method based on the transformation of variable technique as developed by Tay and Kingsbury [1996] and Tay [1998] has been extensively studied in this chapter. The major motivational aspects of the work covered here are based on the use of IIR filters for the design of perfect reconstruction 2-channel quadrature mirror filter bank. This design procedure is simple and it offers vast flexibility for fine-tuning of the overall system. Automating the

optimisation process of the design of the QMF bank has further extended the work reported by Tay [1998]. This was achieved by using genetic algorithms followed by the standard optimisation methods such as the gradient based quasi-Newton and non-gradient based downhill Simplex methods in a hybrid approach. GAs were also used in the second stage for optimisation of IIR filter coefficients based on finite word length constraints. This second stage GA optimisation is useful in the implementation of the system for real-time applications.

The general behaviour of the genetic algorithm as an optimisation tool has been investigated extensively for this application. This has involved developing a special 'creep' code to study the effects of efficiency and robustness of the GA as a stand-alone optimisation tool. The GA 'creep' code mimics the characteristics of a downhill Simplex type tumbling behaviour searching for the optimal minima point. A comparison was drawn with the standard optimisation methods such as quasi-Newton and Simplex. The results for design examples 1, 2 and 3 are shown in Tables 4.1, 4.2 and 4.3 respectively. The outcome evident from the results show that while the 'creep' code has generated good minima values, the hybrid approach is efficient and generates best results consistently. Furthermore, for this application, there is no distinction in the hybrid optimised results using the sequential programming (`constr.m`), the quasi-Newton and the downhill Simplex methods. For completeness of this part of the study, a brief investigation of the execution times for each method was conducted using a relatively slow computer (i.e. 200 MHz Pentium PC). This is shown in Table 4.10. The execution of the main GA code over 20 generations takes approximately 2 minutes. The 'creep' code following this over 80 generations takes substantially longer and is, therefore, not included here for comparison. The results given in Table 4.10 show that the quasi-Newton method is effective and the most efficient optimisation option for this application.

Table 4.10 Execution time in seconds

GA (20 gens.)+	Design example 1	Design example 2	Design example 3
Constr.m	<1	1.47	1.25
Simplex	<1	4.42	1.33
Quasi-Newton	<1	1.03	0.87

Further contributions are in deriving the IIR filter coefficients of the QMF bank and testing the system using the Simulink toolbox of Matlab. The overall amplitude distortion and the error signal for the optimised QMF bank are shown in Figures 4.12(a) and 4.14 respectively. The amplitude distortion is almost entirely flat and the error signal is of the order of the software mathematical error limits. These outcomes clearly indicate the perfect reconstruction characteristic of the design as embedded in the theoretical considerations.

The deduction from the coefficient values of the individual IIR filters as seen in Table 4.5 confirms the possibility of polyphase decomposition of the IIR filter transfer functions. This leads to a computationally efficient structure of the QMF bank. The implicit nature of the transformation function $Z=M(z)$ of Equation 4.14 generates denominator terms of $H_T(Z)$ and $F_T(Z)$ of Equation 4.11 that are both functions of z^2. This property leads to the realisation of the polyphase form of the QMF bank as shown in Figure 4.20. This is significant here because although all linear phase FIR filters can be realised in a polyphase structure form of quadrature mirror filter bank, not all IIR filters can be decomposed directly into polyphase structures. It is for this reason that IIR filters are not always the best option for multirate filter bank applications. The advantage of using fewer number of filter coefficients for the case of IIR filters is lost when computationally efficient polyphase structures of linear phase FIR filters using larger number of filter coefficients for similar magnitude responses are used.

The next stage of study was the implementation of the optimised QMF bank using a real time digital signal processor starter kit (DSK50) [Texas Instruments, 1996]. This is based on the Texas Instruments' TMS320C50 fixed point device. A new proposed structure for real-time test used for this study is shown in Figure 4.26. A process for companding was applied to the lower and the upper frequency segments of the speech signal. Tests were conducted using a small sample of individuals based on variable number of bits for encoding the input signal and the application or absence of the companding scheme. The results are compared using the mean opinion score (MOS) measure and are shown in Table 4.9. These results clearly show an improvement of the MOS measure by using the companding technique as proposed. The best results are for 7-5 encoding (i.e. 7 bit encoding for the lower half of the frequency segment and 5 bit encoding for the upper half of the frequency segment) with companding. This represents a data rate of 48 kbps assuming a sampling rate of 8 kHz. The compression achievable for this form of signal is 4/3.

The major contributions of this chapter are summarised in the following.

- A real-valued genetic algorithm code has been developed for the optimisation of the design of a class of quadrature mirror filter bank that has a perfect reconstruction property.
- This code was further enhanced to include a 'creep' code option within the main GA code that uses a 'tumbling-like' minimisation algorithm. The new 'creep' code was developed to draw a comparative study with the standard quasi-Newton and Simplex optimisation methods. The new GA hybrid optimised results show a significant improvement when compared with the original design results as seen in Tables 4.1, 4.2 and 4.3. However, the most efficient and optimal results are obtained when a hybrid form of GA followed by either quasi-Newton or Simplex methods is used.
- The new GA optimised design of the QMF bank was implemented on a real-time TMS320C50 digital signal processing starter kit. This realisation was in a computationally

efficient polyphase decomposition form. The FWL quantised coefficients were optimised using the genetic algorithm code that has been previously developed in Chapter 3.

- Tests were conducted using the Mean Opinion Score metric that showed an improvement of results (shown in Table 4.9) using the 'with companding' option as proposed in Figure 4.26.

4.10 Summary of Chapter 4 and further comments

This chapter considers the use of genetic algorithms as an optimisation tool in the design of a 2-channel quadrature mirror filter bank. The specific class of perfect reconstruction filter bank using transformation of variables technique developed by Tay and Kingsbury [1993, 1996, 1998] has been investigated. It has been shown that a hybrid process applying GAs followed by a standard optimisation technique such as quasi-Newton or downhill Simplex yield better solutions than the constrained optimisation procedure used by itself as reported in [Tay, 1998]. For real-time implementation, the transformation of variables design technique develops IIR filters (analysis and synthesis) of the form that can be executed in a computationally efficient polyphase structure. The filter coefficients were further optimised using GA's based on finite word length constraints. The analysis and synthesis filters that were optimised using GA's were translated into actual real-time codes for the Texas Instruments' TMS320C50 digital signal processor using a number of Matlab script files developed to aid in coefficient scaling and format transformation. Consideration was given to looking at the aspects of companding to improve the performance of the QMF banks designed. Several real-time tests were conducted using various encoding combinations of the sub-bands. The final results show that sub-band compression with companding has improved the performance of the two-channel QMF bank compared with the simple two-channel QMF bank, for the same bit rates.

Further reduction in data rate can be achieved by encoding the sub-bands with adaptive differential pulse code modulation (ADPCM). For speech it has been shown that differences from one sample to the next are small, therefore a very small number of bits can be used for transmission. In ADPCM the sample differences between consecutive samples is encoded thus allowing improved coding efficiencies to be used. A weighting procedure is used to cover larger sequential changes. For the two-channel QMF bank, Crochiere [1981] proposed a 4-bit ADPCM representation for the lower frequency channel and 2-bit ADPCM representation for the higher frequency channel. Sub-band compression can thus be used in the above schemes to increase the quality of the transmitted speech without increasing the bit rate. Higher and more efficient coding compression is achievable by using a multiple M-channel filter bank of uniformly spaced frequency band segments. The design and optimisation issues of these types of multirate filter banks will be considered in the next chapter.

Chapter 5: Optimisation of a class of M-channel uniform filter bank

Overview of Chapter 5: This chapter deals with the design and optimisation issues relating to a class of M-channel uniform multirate filter bank. The specific case of a cosine modulated pseudo QMF bank is considered for which all the filter bank channels are of equal width on the frequency scale. The design and optimisation of the filter bank is based on the use of a single prototype filter. The optimisation process uses a genetic algorithm technique and the results are compared with the optimised results using the quasi-Newton and downhill Simplex methods. Tests and results of a hybrid GA approach are also included.

5.1 Introduction

The issues of design and optimisation considered in Chapter 2 for finite word length FIR digital filters and for multirate quadrature mirror filter banks considered in Chapter 4 are extended to the case of uniform multiple-band filter banks that is studied in this Chapter. The prototype filters used for the design of the uniform filter banks considered here are based on the use of FIR analysis and synthesis filters. The real-time application of these filter banks on fixed-point devices using finite word length constraints to the specific areas such as telephone speech signal coding and compression can then be implemented.

The application of uniform 2-channel filter banks, as discussed in chapter 4, is sometimes not sufficient for real signals. The choice then is to use a multiple number of frequency bands so that better resolution is obtained. An M-channel uniform filter bank consists of low-pass, band-pass and high-pass filters partitioning the frequency spectrum into equal widths and equidistant from their centre frequencies. In general, the design and practical realisation of M-channel

uniform and non-uniform filter banks for real-time applications with perfect reconstruction property, is a difficult problem to solve. However, theoretical design issues for uniform filter banks with perfect reconstruction property, has now been resolved [Vaidyanathan, 1987(a), 1987(b)]. In these, the lattice structure combinations are given that lead to paraunitary transfer matrices of the analysis filters. The coefficients are numerically optimised such that the transfer functions of each channel give optimal selectivity. Filter banks developed using this technique are robust and immune to quantisation constraints of filter parameters. However, a major drawback is the complexity of design for real-time applications that increases significantly with increased number of channels.

A more acceptable and practical method of design is based on the use of cosine modulation for which all the M-channel filters are derived from a single prototype. Two significant advantages of this method of design are

- The analysis and synthesis filters are of equal lengths and the overall design of the filter bank is based on the design of just one prototype filter plus modulation overhead.
- During the design phase, the number of filter parameters required for optimisation is small.

The closed form approximate solution for designing M-channel uniform filter bank based on cosine modulation technique for which only the directly adjacent aliasing spectra are compensated for by an appropriate mechanism, is considered here. Due to the approximate nature of reconstruction of such filter banks, these are commonly referred to as pseudo QMF systems.

5.2 Maximally decimated uniform filter bank

The structure of a maximally decimated M-channel uniform filter bank is shown in Figure 5.1. The input signal is assumed to have a bandwidth of π radians with respect to the sampling rate at the input and output of the filter bank. The bandwidth of each of the filters is π/M radians. Since the sampling rate of each sub-band signal is reduced by a factor of M, then such a structure is referred to as a maximally decimated filter bank [Vaidyanathan, 1993].

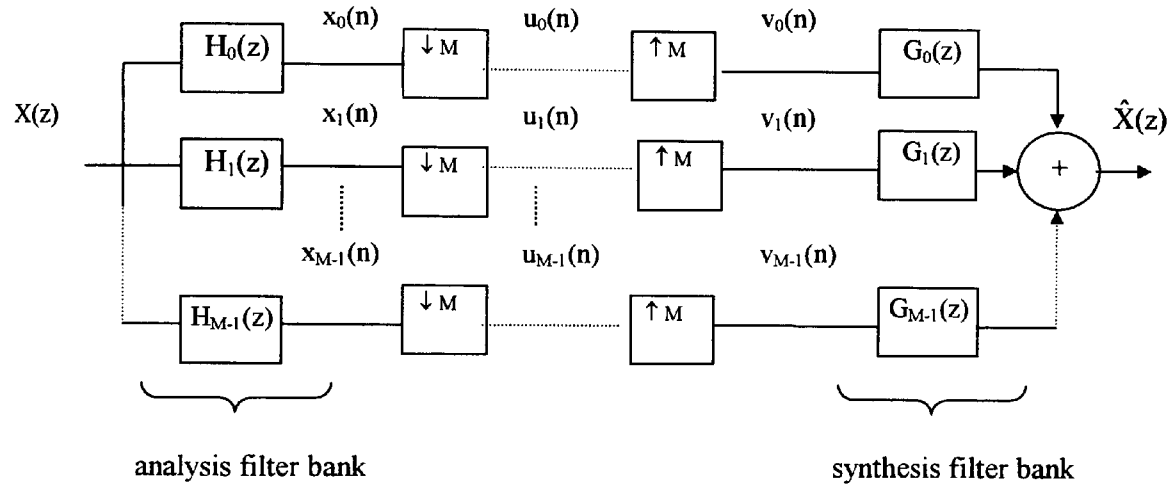

Figure 5.1 M-channel maximally decimated uniform multirate filter bank.

The analysis of the expression for $\hat{X}(z)$ in terms of X(z) will assume the absence of quantisation and coding errors. Each sub-band signal through the analysis filter bank is given by

$$X_k(z) = H_k(z)\, X(z) \qquad 5.1$$

where k = 0,1,2,....,M-1.

The decimated signals $u_k(n)$ have z-transforms given by

$$U_k(z) = \frac{1}{M} \sum_{\ell=0}^{M-1} H_k(z^{1/M} W^{\ell}) X(z^{-1/M} W^{\ell}) \qquad 5.2$$

where $W=W_M = e^{-j2\pi/M}$.

The output of the expanders are given by

$$V_k(z) = U_k(z^M) = \frac{1}{M}\sum_{\ell=0}^{M-1} H_k(z\,W^{\ell})X(z\,W^{\ell}) \qquad 5.3$$

and the reconstructed signal is given by

$$\hat{X}(z) = \sum_{k=0}^{M-1} G_k(z)\,V(z) \qquad 5.4$$

or $$\hat{X}(z) = \frac{1}{M}\sum_{\ell=0}^{M-1}\left[\sum_{k=0}^{M-1} G_k(z)\,H_k(z\,W^{\ell})\right]X(z\,W^{\ell}) \qquad 5.5$$

For simplicity $\hat{X}(z)$ can be written as

$$\hat{X}(z) = \sum_{\ell=0}^{M-1} A_{\ell}(z)X(z\,W^{\ell}) \qquad 5.6$$

where

$$A_{\ell}(z) = \frac{1}{M}\sum_{k=0}^{M-1} G_k(z)\,H_k(z\,W^{\ell}) \qquad \text{for } 0 \le \ell \le M\text{-}1 \qquad 5.7$$

The term $X(zW^{\ell})$ for $\ell \neq 0$ represents the shifted version of the spectrum $X(z)$. The reconstructed output is, therefore, a linear combination of the input signal and its M-1 uniformly shifted aliasing components. For $\ell = 0$, Equation 5.5 can be written as

$$\hat{X}(z) = \frac{1}{M}\sum_{k=0}^{M-1} G_k(z)\,H_k(z).X(z) \qquad 5.8$$

This represents the transfer function of the filter bank. Also since, for perfect reconstruction, no distortions are expected i.e. $\hat{X}(z) = X(z)$, then for a practical system the distortion function is given by the linear distortions of the filter bank and represented by

$$A_0(z) = \frac{1}{M}\sum_{k=0}^{M-1} G_k(z)\,H_k(z) \qquad 5.9$$

Note that when $|A_0(z)|$ is not an all pass condition then there will be amplitude distortion and if $A_0(z)$ has non-linear phase, then there will be phase distortion.

The aliasing components of the input signal must normally be worked out individually for ℓ=1,2,...,M-1 for which the components do not cancel each other. However, in practice an

aliasing function is defined that describes the overall summing of uncorrelated aliasing components in the filter bank. This is given by

$$A_{alias}(z) = \sqrt{\sum_{\ell=1}^{M-1}\left|\frac{1}{M}\sum_{k=0}^{M-1}G_k(z)\,H_k(z\,W^{\ell})\right|^2} \qquad 5.10$$

For perfect reconstruction, the choice of analysis/synthesis filters must be such that aliasing is completely cancelled and $A_0(z)$ is a pure delay. The system is then free from aliasing, amplitude and phase distortions.

5.3 Cosine modulated M-channel pseudo QMF bank

The theory of pseudo QMF systems based on cosine modulation design has been extensively reported in literature [Rothweiler, 1983], [Chu, 1985], [Cox, 1986], [Nguyen, 1992] and [Vaidyanathan, 1993]. Further developments of this form of design methodology have been reported in [Lin and Vaidyanathan, 1995], [Goh and Lim, 1998] and [Agenti and Del Re, 2000]. The procedure for designing these types of filter banks can be summarised by the following

- The filter bank channels are all of equal width and formed by shifting the lowpass prototype to equidistant frequency shifts. The transfer functions of adjacent channels are approximately power complementary between their centre frequencies. This condition leads to the distortion function being approximately a delay.
- The alias cancellation applies only to the directly adjacent channels and all other alias spectra components are assumed to be suppressed by the high stop-band attenuation of the prototype filter.

The approximate nature of the design of such filter banks is the reason for the term 'pseudo' QMF bank.

5.3.1 Design considerations

The key design requirement of a cosine-modulated pseudo-QMF filter bank is to obtain an appropriate transfer function of the prototype filter that will generate approximately power complementary frequency responses of frequency-shifted replicas about the centre frequencies. Also that the stop-band attenuation of the prototype filter is sufficiently large so that all alias spectra, apart from the directly adjacent one, are suppressed. A commonly used prototype filter in this application of the filter bank is approximated by the square root raised-cosine characteristic [Fliege, 1994]. The frequency response for such a low pass filter is given by

$$P(j\omega) = \begin{cases} 1 & \text{for} \quad \dfrac{|\omega|}{\omega_c} \le 1-r \\ \cos\left[\dfrac{\pi}{4r}\left(\dfrac{\omega}{\omega_c} - (1-r)\right)\right] & \text{for} \quad 1-r \le \dfrac{|\omega|}{\omega_c} \le 1+r \\ 0 & \text{for} \quad \dfrac{|\omega|}{\omega_c} \ge 1+r \end{cases} \qquad 5.11$$

Where ω_c is the cut-off radial frequency and 'r' is the roll-off factor that lies in the range $0<r\le1$. For $r\to0$, the ideal brick-wall type of low pass filter is approximated with a cut-off frequency ω_c. Taking the inverse Fourier Transform of the transfer function of Equation 5.11 gives the impulse response p(n) as shown below

$$p(n) = \frac{4rn.\cos\left[n\pi\dfrac{(1+r)}{M}\right] + M.\sin\left[n\pi\dfrac{(1-r)}{M}\right]}{M\left[1-\left(\dfrac{4rn}{M}\right)^2\right].n\pi} \qquad 5.12$$

Where M is the number of channels of the filter bank and $\pi/2M$ is the bandwidth of the prototype filter.

The special condition for $n\to0$ is given by

$$p(0) = \frac{I}{M} + \frac{r}{M}\left(\frac{4}{\pi} - 1\right) \qquad 5.13$$

and for $n = \pm M/4r$ when this is an integer value then the corresponding limits are

$$p(\pm M/4r) = -\frac{r}{M}\left[\frac{2}{\pi}\cos\left(\frac{\pi}{4r}(1+r)\right) - \cos\left(\frac{\pi}{4r}(1-r)\right)\right] \qquad 5.14$$

For a realisable causal filter, the impulse response must be symmetrically truncated and appropriately time-shifted. The causal impulse response that is obtained by convolving with itself results in the M^{th} band filter. The resulting raised-cosine characteristic is then approximately obtained. The overlapping of alias components of the spectra in the design of analysis and synthesis filter banks can be minimised by properly selecting the weighting factors that cause a small shift in the corresponding frequency responses. The closed form expressions for the impulse response of the analysis and synthesis filter banks [Fliege, 1994] are given by

for analysis filter bank

$$h_k(n) = 2\, p(n) \cos.[(k+1/2)(n-(N-1)/2)\pi/M + (-1)^k\pi/4] \qquad 5.15$$

for synthesis filter bank

$$g_k(n) = 2\, p(n) \cos.[(k+1/2)(n-(N-1)/2)\pi/M - (-1)^k\pi/4] \qquad 5.16$$

where $k = 0,1,2,\ldots,M-1$ and N is the number of coefficients used for the prototype filter.

5.3.2 Design examples

An 8-channel uniform filter bank is considered here based on the cosine modulation design using a square root raised-cosine prototype filter. The bandwidth of the prototype filter must be $\pi/16$ in this example for M=8 (note that: bandwidth of the prototype filter is half the bandwidth of individual channels). Three design examples are considered in this section based on FIR filter lengths (i.e. number of coefficients) of N=39, 141 and 257. The higher order filter is normally expected to perform significantly better, both in terms of overall linear distortion and aliasing error, when compared to the lower order filter. However, there are clear issues about

real-time implementation and computational overheads, that become significant when larger number of filter coefficients are used.

Due to the approximately complementary frequency responses of the frequency-shifted replicas of the prototype filter for the cosine modulated design, an optimisation process based on marginally changing the specifications of the prototype filter can be expected to generate improved results. This conjecture was applied to test by using two variables i.e. the bandwidth of the prototype and the roll off factor 'r'. The bandwidth of the prototype filter was assumed to be some value π/M_p, where M_p is a real valued number approximately equal to 2M (for an 8 channel system $M_p \cong 16$). Due to the variation in the value of M_p, the restriction on 'r' i.e. $0<r \leq 1$ no longer applies. The roll off value 'r' is thus allowed to vary in the range 0 to 2. The characteristic of the square root raised cosine prototype filter with $r \cong 0.5$ could offer sufficient stop band separation for higher order filters but for lower order filters 'r' may be closer to 1 to allow for adequate aliasing error cancellation.

5.3.3 Peak distortions

Two types of distortions are considered here to represent the quality of the output signal of the filter bank. The first is the amplitude distortion given by Equation 5.9 and represented by the peak to peak ripple of $M|A_0(z)|$ i.e.

$$E_{pp} = \max[M|A_0(z)|] - \min[M|A_0(z)|] \qquad 5.17$$

The second is the aliasing distortion that is derived by taking the maximum value of $A_{alias}(z)$ (Equation 5.10) over all ω. This gives the worst possible peak aliasing distortion i.e.

$$E_A = \max[A_{alias}(z)] \qquad 5.18$$

5.3.4 Simulink tests

Each of the optimised design examples was tested using the Simulink toolbox of Matlab version 5.3. A typical test configuration is shown in Figure 5.2. The input applied to the filter bank is a random signal with uniform distribution in the range 1 to −1. The error signal v_e is then obtained from the difference between the output signal and an appropriately delayed version of the input signal. The root mean square value of the error signal is then calculated using

$$v_{rms} = \sqrt{\frac{\sum_{1}^{K} v_e^{\,2}}{K}} \qquad 5.19$$

Where K is the number of sampled points of the error signal. A value of K=1000 was used for the tests conducted and reported here. Also, the maximum peak to peak magnitude of the output error signal is derived and is represented by out_{pp}.

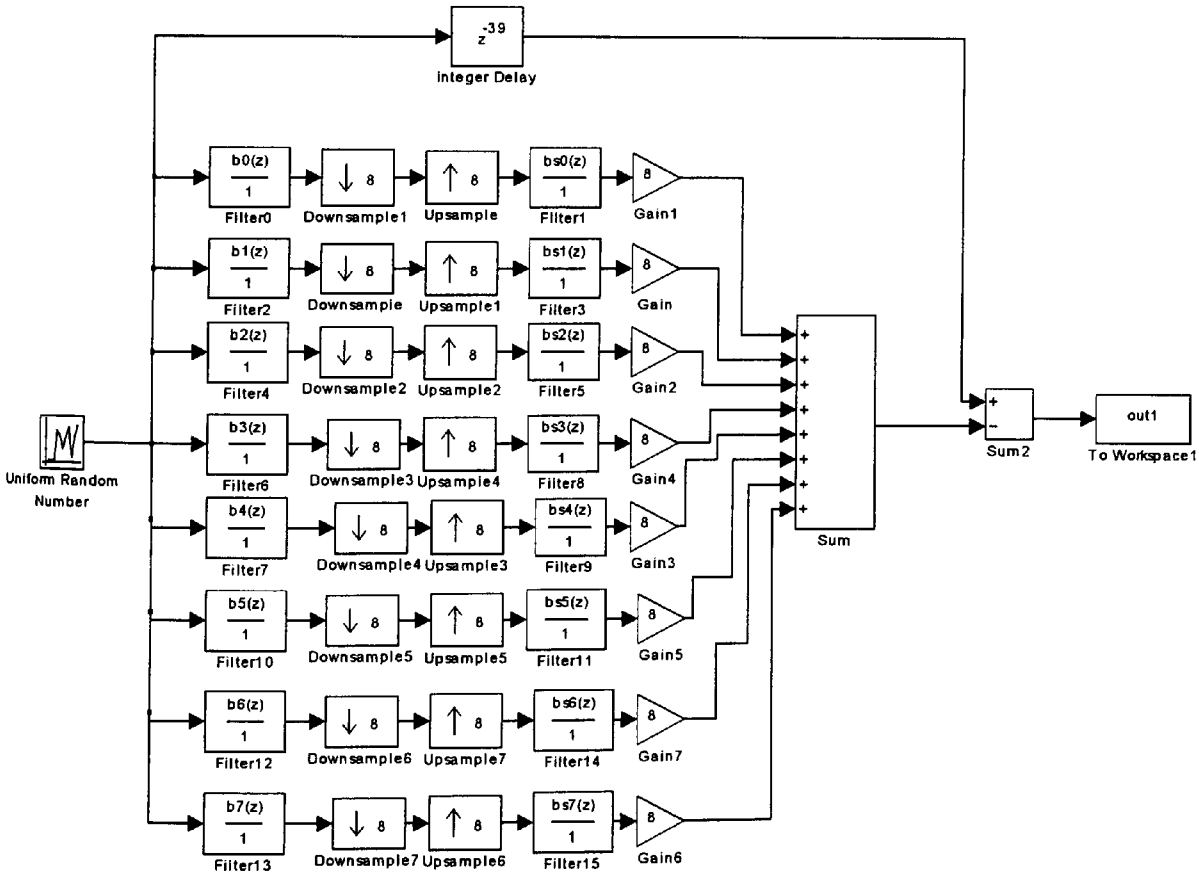

Figure 5.2 A Simulink test circuit for an 8 band uniform filter bank.

5.3.5 Optimisation methods

Three optimisation methods are used in the present study. The first method is based on using the downhill Simplex algorithm due to Nelder and Mead [1965] and has been implemented as the `fmins.m` function of the optimisation toolbox of Matlab. The second method is a gradient based unconstrained quasi-Newton method implemented as `fminu.m` function of Matlab. The third method is based on a simple real-valued genetic algorithm developed as a Matlab toolbox by Chipperfield et al [1993]. A GA code developed for the present work based on this toolbox and used for the optimisation process is shown in Appendix E1.1. The choice for the selection of these three methods is based on the unknown nature of the landscape profile of the function under test. Genetic algorithm is efficient at conducting an extensively parallel search over a wide landscape that may be highly discontinuous. However, it may be less efficient at optimising a potential minima point. The Simplex algorithm is a non-gradient based search method that optimises using a 'tumbling' action and performs well in a highly discontinuous environment although it may be slow and becomes less efficient than the gradient based methods for problems of order greater than two. The unconstrained gradient based quasi-Newton method works well in a continuous environment but is susceptible to getting polarised within a local minima point. The starting 'seed' values are thus crucial towards identifying a more global minima point both for the Simplex and the quasi-Newton algorithms. Some 'trial-and-error' attempts for the starting 'seed' values, therefore, become inevitable in such cases.

5.3.6 GA optimisation methodology and pseudo code

The genetic algorithm used here for the design of the uniform M-channel filter bank is identical to the generic form explained in section 1.4, Chapter 1. This is a Matlab based algorithm developed originally for control systems applications [Chipperfield et al, 1993]. The main GA

code has been adapted for the application in this work and new functions have been written for working out the error objective function. The specific steps followed for the design stage of the QMF bank GA optimisation are shown here and the pseudo GA code is shown in Figure 5.3.

1) Define the GA parameters

The GA parameters used for the design examples i.e. N=39, 141 and 257 are given by

```
GGAP=0.8;       % generational gap
INSR=0.8;       % reinsertion rate
MAXGEN=10;      % number of generations
Nind=100;       % population size
MutRate=0.1;    % mutation rate
```

2) Create population set of individuals

The starting set of parameter values of prototype filter bandwidth is defined in the bounds Mp= 13 to 19, the roll-off factor is defined in the bounds r= 0 to 2 and the trade-off parameter is defined in the bounds α= 0.1 to 0.9. The bounded parameter values are described in a matrix 'FieldDR' and an initial population set consisting of random real-valued individuals is created within the bounds specified in FieldDR matrix. The function **crtrp** of the GA Matlab toolbox is used for this purpose.

3) The Objective function evaluation

The main purpose of the optimisation process here is to minimise the objective function with the specific aim of minimising the overall magnitude and aliasing error so that a perfect reconstruction characteristic is closely met. The error objective function to be minimised and used in the optimisation process is based on the multiple objectives of linear distortion (E_{pp}) and maximum aliasing distortion (E_A) and is given by

$$Obj\_err = \alpha E_{pp} + (1-\alpha)E_A \qquad 5.20$$

Where α is the trade-off parameter given by $0 \leq \alpha \leq 1$. Note that E_{pp} and E_A are defined by Equations 5.17 and 5.18 respectively.

4) Fitness value and ranking

The Matlab based **ranking** function of the GA toolbox ranks the individuals according to their objective function values 'Obj_err' and returns a column vector consisting of the corresponding fitness value 'FitnV' of the individuals. This function performs a linear ranking with a selective pressure (SP) of 2 [Whitley, 1998]. The fitness value assigned to the individuals is calculated according to the formula given by Equation 1.1 in Chapter 1.

5) Selection of individuals for breeding

The high-level function for selection of individuals from the population set and returning the selected individuals in a new population is performed by the **select** function. The low-level selection function **sus** is called by the **select** function. The **sus** function is based on a form of stochastic sampling method and is implemented by obtaining a cumulative sum of the fitness vector 'FitnV' and generating a set of equally spaced numbers between 0 and Σ(FitnV) [Baker, 1987]. The probability of an individual being selected is given by Equation 1.2 in Chapter 1.

6) Recombining individuals – crossover

The crossover function is also performed in two stages. The high-level function is **recombin** that calls the low-level function **recdis**. The **recdis** function is a discrete recombination function. The mating process is performed between pairs of rows. The **recdis** function first generates an internal mask table that determines which parents contribute which variables to the offspring. On the basis of the randomly generated mask table, the variable values are exchanged between the individuals and return a new population after mating.

7) Mutation

The **mutbga** function of the Matlab GA toolbox takes real-valued population, mutates each variable with given probability and returns the population after mutation. The **mutbga** function produces firstly a random internal mask table that determines which variables will mutate and also the sign for the step size. A second internal table generates the normalised mutation step size. The mutated variable is worked out as a function of the original variable and the step size [Muhlenbein and Schlierkamp-Voosen, 1993].

8) Reinsert offspring into new population

The new population set generated after crossover is subjected to the objective function evaluation of each new individual. On the basis of their fitness, the offspring are selected for reinsertion using the **reins** function into the new population. The objective function values are then copied to the reinserted offspring and the GA loop is then repeated for the next generation.

Note that the design of the closed form uniform filter bank based on cosine modulation method eliminates phase distortion and also eliminates aliasing approximately. So it merely remains to reduce amplitude distortion for which $\alpha=1$ in Equation 5.20 [Vaidyanathan, 1993]. However, for lower order prototype filters (such as N=39) the effect of aliasing distortion may contribute significantly towards the overall distortion. The appropriate choice for α then becomes an important consideration in the optimisation process of the multiple objective error function. Such a method for optimising a combination of objectives can lead to producing a single compromise solution that may need no further interaction with the decision-making strategy. However, in the application considered here, the choice for α can be arbitrarily selected thus some fine-tuning of the aggregating function is required in order to obtain a good compromise solution. A simplistic analysis was conducted for this purpose based on the design example 1

(i.e. N=39, M=8) and the results are shown in section 5.4. A more generalised multiple objective optimisation process involves consideration of a vector of objectives that must be traded off in order to obtain the best compromise solution. On this issue, a concept of non-inferiority or Pareto optimality is used to characterise the objectives [Censor, 1977], [Zadeh, 1963] and [Fonsesca and Fleming, 1998]. These considerations are beyond the scope of the present work being studied here. However, methods for categorising landscapes in order to identify the most effective decision making strategy is an important area that has yet to be fully investigated.

```
% Pseudo GA code for optimisation of a M-Channel uniform filter bank
% GA characteristics

FieldDR = [13 0 0.1;     % variables: Mp, r and alpha: lower bound
     19 2 0.9];          % upper bound

% create initial population within the bounds specified in FieldDR
matrix.
Chrom = crtrp(Nind, FieldDR);

% calculate objective function
ObjVal = ufga_obj(Chrom);
[Best(gen+1),ix] = min(ObjVal);

% generational loop
while gen < MAXGEN
   % Fitness
   FitnV = ranking(ObjVal);
   % Selection
   SelCh = select('sus',Chrom,FitnV,GGAP);
   % crossover
   SelCh = recombin('recdis', SelCh, 1);
   % mutation
   SelCh = mutbga(SelCh,FieldDR, MutRate);
   % calculate objective function
   ObjVOff = feval('ufga_obj',SelCh);
   % reinser best individuals
   [Chrom, ObjVal] = reins(Chrom, SelCh,1,1, ObjVal, ObjVOff);
   gen = gen + 1
   [Best(gen+1,1),ix] = min(ObjVal)
   acbest = Chrom(ix,:);
end
```

Figure 5.3 A Pseudo GA code for optimisation of an M-Channel uniform filter bank.

```
% objective function for 8-channel uniform filter
function f = ufga_obj(Chrom);
[Nind,Nvar]=size(Chrom);
for irun = 1:Nind;
   ac = Chrom(irun,:);

a0=ac(1); a1=ac(2); a2=ac(3);

N=141;       % number of filter coefficients
Mp=a0;       % bandwidth variable for the prototype filter
r=a1;        % roll off value
alpha=a2;    % trade-off parameter

% define prototype filter
% derive analysis filters
% derive synthesis filters
% derive distortion function
% derive aliasing fuction
% calculate the objective function
err = alpha*(max(T)-min(T))+ (1-alpha)*max(Tal);

f(irun,:) = err;
end
```

Figure 5.4 Pseudo objective function code for optimisation of an M-Channel uniform filter bank.

5.4 Some results

A number of tests were conducted based on a standard design method using the square root raised cosine characteristic of the prototype filter derived using Equation 5.12 and the closed form expressions for the analysis and synthesis filters as given by Equations 5.15 and 5.16. Some comparisons are also made with the results of other researchers. Vaidyanathan [1993, pp. 336] lists the coefficients of a FIR low pass prototype filter used in the design of an 8 channel pseudo QMF bank by using the closed form expressions of Equations 5.15 and 5.16. However, the coefficients were optimised using a non-linear optimisation package [Press et al, 1989] based on the minimisation of a composite objective function formed using a 'trade-off' parameter between the linear distortion and the stop band attenuation of the prototype filter. This design was reconstructed using Matlab version 5.3 package and the results so derived were

tested against the optimised results of the square root raised cosine prototype filter. Another comparison was drawn with the graphical results of Fliege [1994, pp. 202] based on the square root raised cosine prototype filter. No reconstruction of this design example was possible since the roll-factor is not specified. However, the optimisation procedures developed in the present work were applied to this example using the same number of coefficients (i.e. N=257) and some comparisons were made based on the visually observable results of Fliege [1994].

5.4.1 Design example 1: For N=39 and M=8

The choice for N=39 in this design example was made so that it offers a comparison with the results given by Vaidyanathan [1993, pp. 336] (note that Vaidyanathan uses one more i.e. N=40 coefficients). Comprehensive test results for the three optimisation methods i.e. Simplex, quasi-Newton and genetic algorithm, for different values of the trade-off parameter α are shown in Table 5.1 (a), (b) and (c) respectively. Here, the variables M_p, r, E_{pp} and E_A have been defined previously (sections 5.3.2 and 5.3.3). However, 'c' is a compensating factor that scales the optimised FIR prototype filter coefficients with an appropriate rectangular window such that the average gain in the passband of the prototype is unity. Also, v_{rms} and out_{pp} are the root mean square and the maximum peak to peak error values respectively taken from the Simulink test results as discussed in section 5.3.4. Table 5.1 (d) shows the results of the reconstructed 8-channel pseudo QMF bank taken from Vaidyanathan [1993].

Comparing the results of the three optimisation methods, the downhill simplex method results given in Table 5.1(a) show a clear trend in the variation of E_{pp} and E_A against the trade-off parameter α. It must be recognised that the objective function being minimised is a function of the two parameters E_{pp} and E_A. However, when these two parameters are considered in isolation of the Simulink results, they do not convey a full picture of the quality of the system in terms of the overall output response. For this reason, the two parameters, v_{rms} and out_{pp} as

defined in section 5.34, are crucial towards assessing the overall performance of the system. In general, it is also observed that when v_{rms} is a minimum then out_{pp} is also a minimum. This condition forms the basis for the choice of the parameters to be used for the prototype filter in the design of the pseudo-QM filter bank. The results of the quasi-Newton method shown in Table 5.1 (b) behaved somewhat erratically and did not perform as well as the Simplex method. The GA optimised results as shown in Table 5.1(c) performed well, however, the best minima results were not as good as the best results of the Simplex method. Further comparison with the reconstructed results taken from Vaidyanathan [1993, pp 336] and given in Table 5.1(d) show a significant improvement of the new optimised results. This optimisation study was extended further by using the GA optimised results of Table 5.1(c) as starting 'seed' values for the Simplex method. The results of this hybrid approach are shown in Table 5.2. Again the Simulink test results show a certain trend in terms of the trade-off parameter α, however, no significant advantage is evident in comparison with the direct Simplex method results of Table 5.1(a).

The best optimised results taken from Table 5.2 for $\alpha = 0.1$ generate coefficients of the prototype filter as shown in Table 5.3. Note that only the first half of the coefficients are shown due to the linear phase characteristic of the prototype filter. These values were used to plot the frequency responses of the prototype filter and the eight analysis filters of the pseudo-QMF bank as shown in Figures 5.5 (a) and (b) respectively. The linear and aliasing distortion responses are shown in Figures 5.6 (a) and (b) respectively. The Simulink test error signal for a random input with uniform distribution in the range -1 to 1 is shown in Figure 5.7. The reconstructed plots of the optimised coefficients given by Vaidyanathan [1993] are shown in Figures 5.8 (a) and (b) for the prototype filter response and the analysis filter responses respectively. The linear and aliasing distortions are shown in Figures 5.9 (a) and (b) respectively and the Simulink test error plot is shown in Figure 5.10.

5.4.2 Design example 2: For N=141 and M=8

The three optimisation methods as used in the design example 1 are also used in this case. However, the best GA optimised results showed significant improvement over the best Simplex or quasi-Newton method results. Tables 5.4(a), (b) and (c) show the results of the Simplex, quasi-Newton and GA optimisation methods respectively. These results clearly indicate a strong dependency of the optimisation process on the starting 'seed' values for the Simplex and quasi-Newton methods. An interesting outcome of this observation leads to a brief understanding of the objective function profile. While for the case of design example 1, the objective function has a sufficient slope for the Simplex algorithm to 'tumble' towards a global minima point, this does not appear to happen in the case of design example 2. The objective function profile for the design example 2 clearly indicates regions that are sufficiently flat for the Simplex or the quasi-Newton methods to optimise towards local minima points. The genetic algorithm method, however, is seen to be flexible and robust in searching towards a global minima point over a wide landscape. A hybrid approach in this case would be an effective strategy in the search towards a global minima point.

Table 5.1

(a) Optimised results using downhill Simplex method with seed values M_p=16.01 and r=0.51 for N=39

α	M_p	r	c	E_{pp}	E_A	v_{rms}	out_{pp}
0.0100	17.1368	1.2063	1.0355	0.0038	0.0005	0.0012	0.0078
0.1000	17.1540	1.2109	1.0360	0.0037	0.00050	0.0012	0.0078
0.2000	17.1843	1.2189	1.0369	0.0036	0.00053	0.0013	0.0102
0.3000	17.2698	1.2411	1.0396	0.0033	0.0007	0.0013	0.0102
0.4000	17.5029	1.2984	1.0465	0.0025	0.0011	0.0028	0.0325
0.5000	18.5612	1.5090	1.0773	0.0009	0.0025	0.0063	0.0675
0.6000	19.5677	1.6617	1.1056	0.0007	0.0027	0.0069	0.0723
0.7000	19.4841	1.6500	1.1032	0.0007	0.0027	0.0069	0.0724
0.8000	19.4861	1.6503	1.1033	0.0007	0.0027	0.0069	0.0724
0.9000	19.4653	1.6474	1.1027	0.0007	0.0027	0.0069	0.0725
1.0000	19.4258	1.6418	1.1017	0.0007	0.0027	0.0069	0.0725

(b) Optimised results using unconstrained quasi-Newton method with seed values M_p=16.01 and r=0.51 for N=39

α	M_p	r	c	E_{pp}	E_A	v_{rms}	out_{pp}
0.0100		results	did	not	converge		
0.1000		results	did	not	converge		
0.2000	17.2316	1.2313	1.0384	0.0034	0.0006	0.0014	0.0141
0.3000	17.2632	1.2394	1.0394	0.0033	0.0006	0.0015	0.0166
0.4000	18.1130	1.4283	1.0644	0.0013	0.0021	0.0053	0.0582
0.5000		results	did	not	converge		
0.6000	17.7412	1.3523	1.0535	0.0019	0.0016	0.0039	0.0446
0.7000	16.8521	1.1461	1.0263	0.0081	0.0008	0.0024	0.0168
0.8000	18.5781	1.5118	1.0778	0.0009	0.0025	0.0063	0.0677
0.9000	17.7911	1.3631	1.0550	0.0018	0.0017	0.0042	0.0469
1.0000	18.4758	1.4944	1.0749	0.0010	0.0024	0.0062	0.0662

(c) Optimised results using GA with M_p range = 13 to 19 and 'r' range = 0 to 2 for N=39

α	M_p	r	c	E_{pp}	E_A	v_{rms}	out_{pp}
0.0100	17.1570	1.2144	1.0360	0.0041	0.0005	0.0013	0.0101
0.1000	17.4690	1.2870	1.0453	0.0032	0.0010	0.0024	0.0275
0.2000	17.2164	1.2248	1.0378	0.0039	0.0005	0.0013	0.0105
0.3000	17.4292	1.2811	1.0442	0.0028	0.0010	0.0024	0.0282
0.4000	17.4690	1.2870	1.0453	0.0032	0.0010	0.0024	0.0275
0.5000	17.4292	1.2797	1.0442	0.0029	0.0010	0.0023	0.0271
0.6000	17.4565	1.2870	1.0451	0.0027	0.0010	0.0025	0.0295
0.7000	17.8809	1.3820	1.0576	0.0017	0.0018	0.0045	0.0505
0.8000	17.4312	1.2811	1.0443	0.0028	0.0010	0.0024	0.0280
0.9000	17.0328	1.8707	1.0299	0.0006	0.0187	0.0465	0.3184
1.0000	17.0211	1.8672	1.0296	0.0006	0.0186	0.0464	0.3178

(d) Reconstructed results derived using the optimised prototype coefficients taken from Vaidyanathan [1993] for N=40

E_{pp}	E_A	v_{rms}	out_{pp}
0.0108	0.0023	0.0061	0.0406

Table 5.2

Optimised results using hybrid downhill Simplex method with seed values taken from GA results of Table 5.1 (c) for N=39.

α	M_p	r	c	E_{pp}	E_A	v_{rms}	out_{pp}
0.0100	17.1367	1.2063	1.0355	0.0038	0.0005	0.0012	0.0078
0.1000	17.1541	1.2109	1.0360	0.0037	0.0005	0.0012	0.0078
0.2000	17.1846	1.2190	1.0369	0.0036	0.0005	0.0013	0.0102
0.3000	17.2697	1.2411	1.0396	0.0033	0.0007	0.0013	0.0102
0.4000	17.5025	1.2983	1.0465	0.0025	0.0011	0.0028	0.0325
0.5000	18.9648	1.5739	1.0888	0.0007	0.0027	0.0068	0.0714
0.6000	19.5060	1.6531	1.1038	0.0007	0.0027	0.0069	0.0724
0.7000	19.6209	1.6690	1.1070	0.0007	0.0027	0.0069	0.0722
0.8000	19.5569	1.6602	1.1053	0.0007	0.0027	0.0069	0.0723
0.9000	17.6164	2.0354	1.0459	0.0000	0.0204	0.0520	0.3411
1.0000	17.6164	2.0354	1.0459	0.0000	0.0204	0.0520	0.3411

Table 5.3

Optimised prototype filter coefficients using parameters taken from Table 5.2 for $\alpha = 0.1$ for design example 1 (N=39). Only the first half of the coefficients are shown.

n	p(n)
0	-1.1829E-04
1	-1.5375E-03
2	-2.9491E-03
3	-4.0909E-03
4	-4.6664E-03
5	-4.3725E-03
6	-2.9317E-03
7	-1.2420E-04
8	4.1815E-03
9	1.0005E-02
10	1.7239E-02
11	2.5645E-02
12	3.4865E-02
13	4.4439E-02
14	5.3841E-02
15	6.2516E-02
16	6.9928E-02
17	7.5601E-02
18	7.9164E-02
19	8.0379E-02

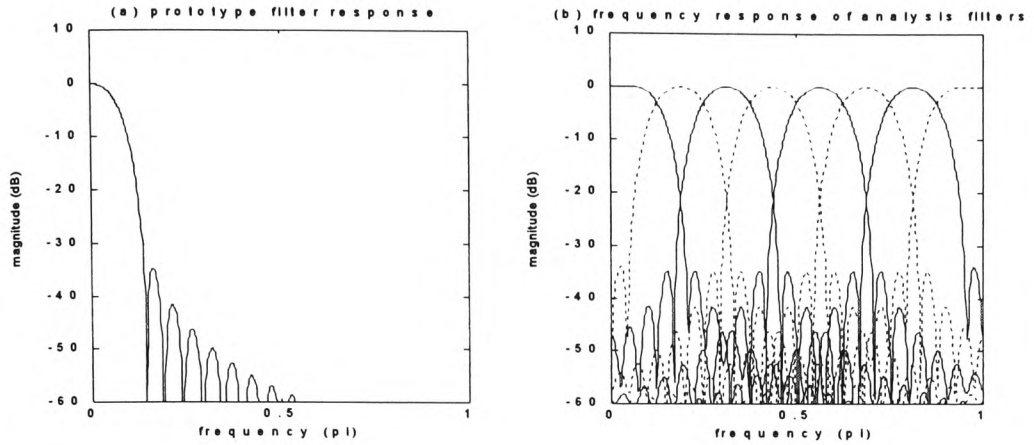

Figure 5.5 Magnitude frequency response for N=39 (Table 5.2 optimised results): (a) prototype filter and (b) all channels.

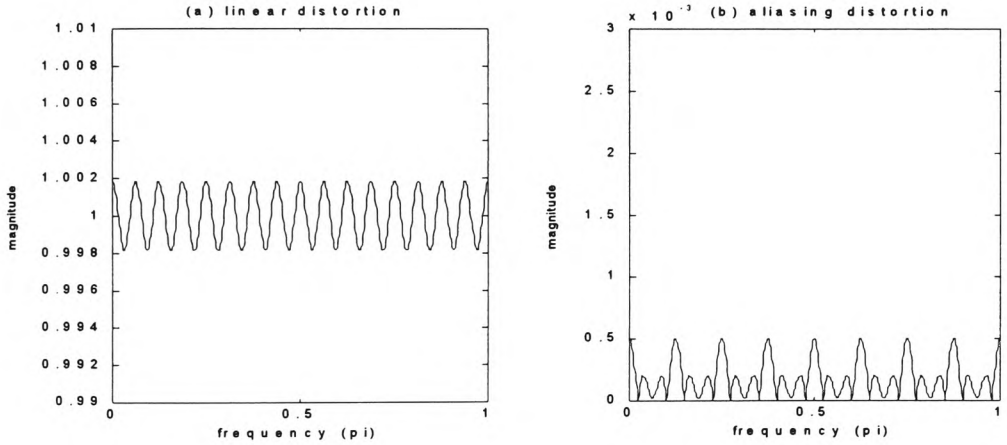

Figure 5.6 Overall distortion for N=39 (Table 5.2 optimised results): (a) linear and (b) aliasing.

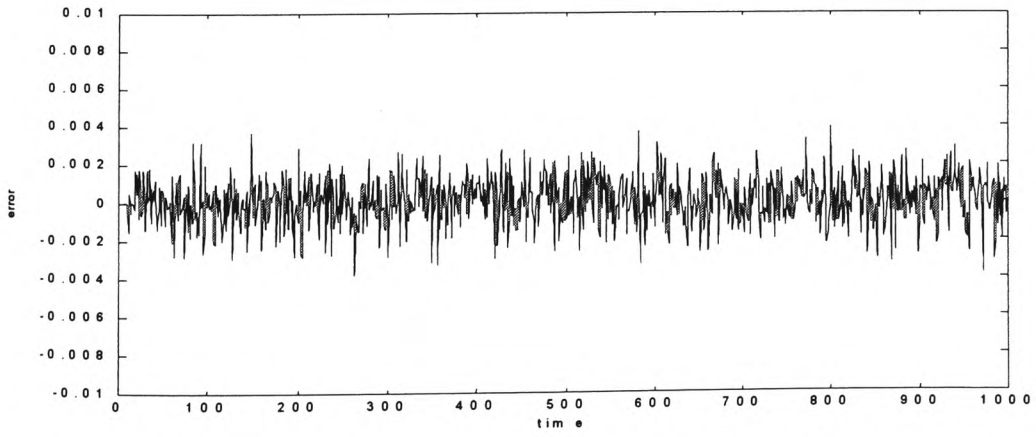

Figure 5.7 Simulink test error signal for a random input with a uniform distribution [-1,1] for N=39 (Table 5.2 optimised results).

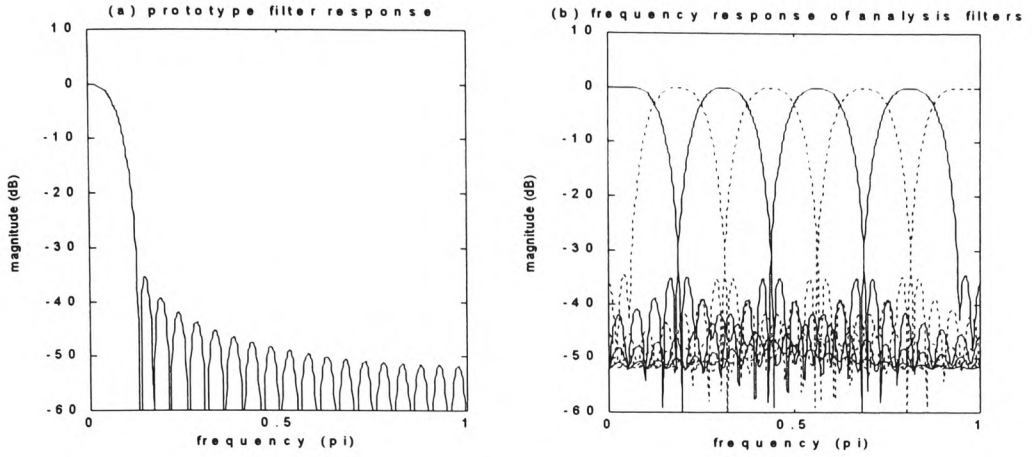

Figure 5.8 Magnitude frequency response for N=40 (Vaidyanathan [1993] optimised results): (a) prototype filter and (b) all channels.

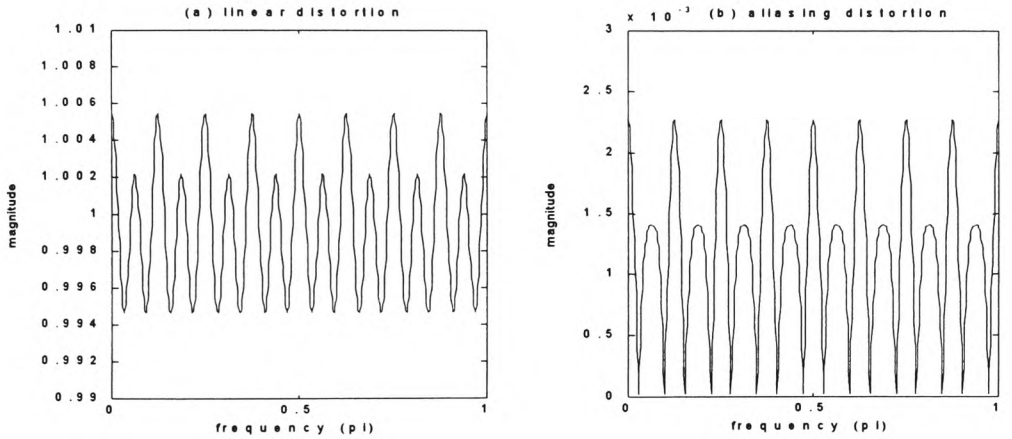

Figure 5.9 Overall distortion for N=40 (Vaidyanathan [1993] optimised results): (a) linear and (b) aliasing.

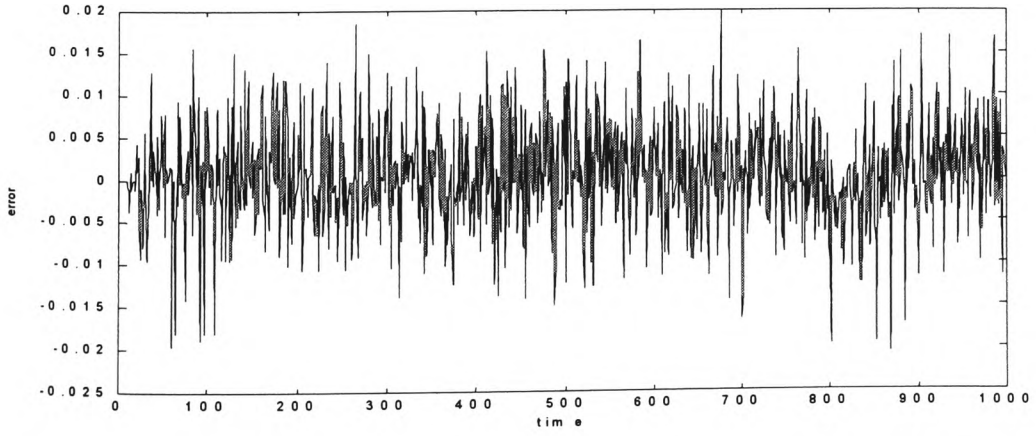

Figure 5.10 Simulink test error signal for a random input with a uniform distribution [-1,1] for N=40 (Vaidyanathan [1993] optimised results).

Due to the dependency of the optimisation process on the trade-off parameter α, the GA search method was further enhanced for this design example by including α as the third variable. The search for the minima value, therefore, includes a range for prototype filter bandwidth π/M_p where M_p varies from 13 to 19 (for an 8 band QMF bank), the roll-off factor 'r' varies in the range 0 to 2 and parameter α varies in the range 0.1 to 0.9. The GA optimised results are shown in shown in Table 5.4(c). These results were then used as starting 'seed' values for the Simplex method and the final optimised results obtained are shown in Table 5.4(d). The hybrid optimised results show a significant improvement over the directly optimised results for the Simplex and the quasi-Newton methods and a marginal improvement over the GA optimised results. The best optimised results taken from Table 5.4(d) generate coefficients of the prototype filter as shown in Table 5.5. Note that only the first half of the coefficients are shown due to the linear phase characteristic of the prototype filter. These values were used to plot the frequency responses of the prototype filter and the eight analysis filters of the pseudo-QMF bank as shown in Figures 5.11 (a) and (b) respectively. The linear and aliasing distortion responses are shown in Figures 5.12 (a) and (b) respectively. The Simulink test error signal is shown in Figure 5.13.

Table 5.4

(a) Optimised results using downhill Simplex method with seed values M_p=16.01 and r=0.51 for N=141

α	M_p	r	c	E_{pp}	E_A	v_{rms}	out_{pp}
0.0100	16.0077	0.4788	1.0000	0.0109	0.0002	0.0017	0.0096
0.1000	15.9993	0.4878	1.0000	0.0103	0.0002	0.0018	0.0121
0.2000	15.9991	0.4880	1.0000	0.0102	0.0002	0.0018	0.0121
0.3000	15.9991	0.4881	1.0000	0.0102	0.0002	0.0018	0.0122
0.4000	15.9992	0.4881	1.0000	0.0102	0.0002	0.0018	0.0122
0.5000	15.9992	0.4881	1.0000	0.0102	0.0002	0.0018	0.0122
0.6000	15.9992	0.4881	1.0000	0.0102	0.0002	0.0018	0.0122
0.7000	15.9992	0.4881	1.0000	0.0102	0.0002	0.0018	0.0122
0.8000	15.9992	0.4881	1.0000	0.0102	0.0002	0.0018	0.0122
0.9000	15.9992	0.4880	1.0000	0.0102	0.0002	0.0018	0.0121
1.0000	15.9992	0.4880	1.0000	0.0102	0.0002	0.0018	0.0121

(b) Optimised results using quasi-Newton method with seed values M_p=16.01 and r=0.51 for N=141

α	M_p	r	c	E_{pp}	E_A	v_{rms}	out_{pp}
0.0100	16.0076	0.4788	1.0000	0.0109	0.0002	0.0017	0.0096
0.1000	16.0039	0.4829	1.0000	0.0106	0.0002	0.0017	0.0104
0.2000	16.0068	0.4798	1.0000	0.0108	0.0002	0.0017	0.0097
0.3000	16.0060	0.4806	1.0000	0.0108	0.0002	0.0017	0.0099
0.4000	15.9992	0.4881	1.0000	0.0102	0.0002	0.0018	0.0122
0.5000	15.9988	0.4877	1.0000	0.0103	0.0002	0.0018	0.0121
0.6000	15.9991	0.4881	1.0000	0.0102	0.0002	0.0018	0.0122
0.7000	15.9988	0.4877	1.0000	0.0103	0.0002	0.0018	0.0121
0.8000	15.9991	0.4881	1.0000	0.0102	0.0002	0.0018	0.0122
0.9000	15.9992	0.4881	1.0000	0.0102	0.0002	0.0018	0.0122
1.0000		results	did	not	converge		

(c) Optimised results using GA with M_p range=13 to 19, 'r' range = 0 to 2 and alpha range= 0.1 to 0.9 for N=141.

α	M_p	r	c	E_{pp}	E_A	v_{rms}	out_{pp}
0.1045	16.0169	0.9015	1.0005	0.0022	0.0002	0.0005	0.0029

(d) Optimised results using hybrid downhill Simplex method with seed values taken from GA results of Table 5.3 (c) for N=141.

α	M_p	r	c	E_{pp}	E_A	v_{rms}	out_{pp}
0.1045	16.0055	0.9064	1.0002	0.0015	0.0002	0.0004	0.0027

Table 5.5

Optimised prototype filter coefficients using parameters taken from Table 5.4(d) for design example 2 (N=141). Only the first half of the coefficients are shown.

n	p(n)	n	p(n)
0	-1.5804E-04	36	-1.3155E-03
1	-2.4737E-04	37	-1.3832E-03
2	-3.0927E-04	38	-1.2679E-03
3	-3.3339E-04	39	-9.6292E-04
4	-3.1408E-04	40	-4.8720E-04
5	-2.5148E-04	41	1.1323E-04
6	-1.5187E-04	42	7.6797E-04
7	-2.7163E-05	43	1.3881E-03
8	1.0639E-04	44	1.8754E-03
9	2.3023E-04	45	2.1338E-03
10	3.2590E-04	46	2.0822E-03
11	3.7767E-04	47	1.6681E-03
12	3.7494E-04	48	8.7859E-04
13	3.1403E-04	49	-2.5026E-04
14	1.9924E-04	50	-1.6290E-03
15	4.2807E-05	51	-3.1152E-03
16	-1.3625E-04	52	-4.5190E-03
17	-3.1426E-04	53	-5.6151E-03
18	-4.6589E-04	54	-6.1590E-03
19	-5.6755E-04	55	-5.9069E-03
20	-6.0083E-04	56	-4.6386E-03
21	-5.5552E-04	57	-2.1796E-03
22	-4.3170E-04	58	1.5782E-03
23	-2.4059E-04	59	6.6598E-03
24	-3.9658E-06	60	1.2997E-02
25	2.4799E-04	61	2.0423E-02
26	4.8012E-04	62	2.8678E-02
27	6.5674E-04	63	3.7424E-02
28	7.4658E-04	64	4.6260E-02
29	7.2755E-04	65	5.4747E-02
30	5.9072E-04	66	6.2444E-02
31	3.4288E-04	67	6.8934E-02
32	7.2684E-06	68	7.3854E-02
33	-3.7798E-04	69	7.6924E-02
34	-7.6341E-04	70	7.7967E-02
35	-1.0939E-03		

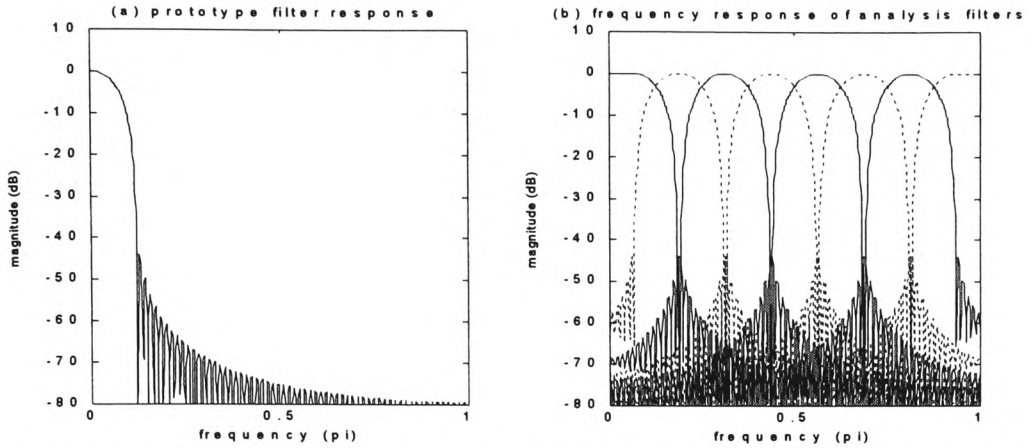

Figure 5.11 Magnitude frequency response for N=141 (Table 5.4(d) optimised results): (a) prototype filter and (b) all channels.

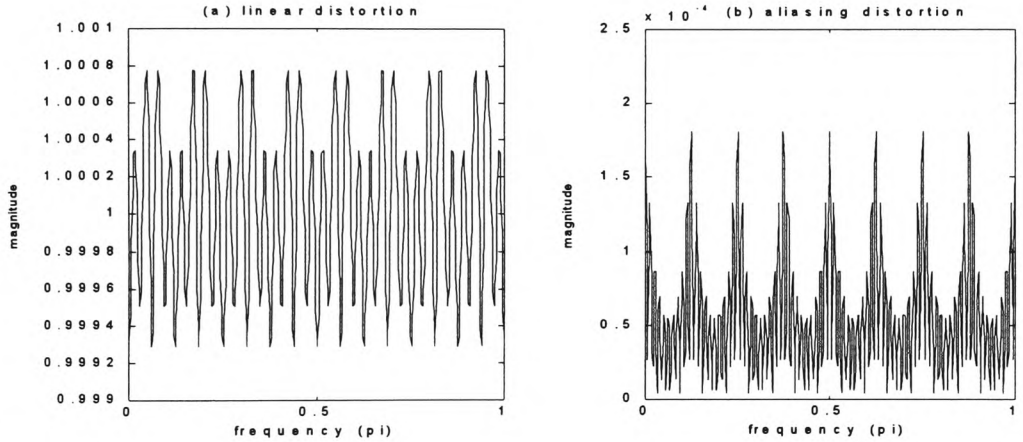

Figure 5.12 Overall distortion for N=141 (Table 5.4(d) optimised results): (a) linear and (b) aliasing.

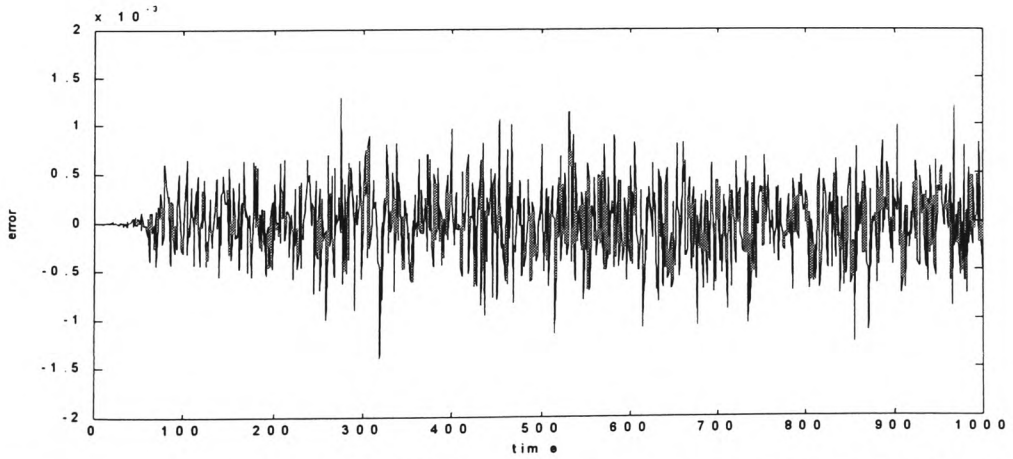

Figure 5.13 Simulink test error signal for a random input with a uniform distribution [-1,1] for N=141 (Table 5.4(d) optimised results).

5.4.3 Design example 3: For N=257 and M=8

The choice for N=257 in this design example was made as it offers a comparison with the graphical results given by Fliege [1994, pp. 202]. However, no prototype filter coefficients are included and no value of the roll-off factor 'r' is specified so a reconstruction of the filter banks was not possible. It is mentioned though that the linear distortion can be reduced by selecting the bandwidth of the prototype to be a value $\pi/15.92$ instead of $\pi/16$. The graphical plots of the linear distortion and aliasing given by Fliege [1994] have been used to estimate the peak to peak linear distortion E_{PP}=0.0085 and the maximum aliasing distortion E_A= 0.0008. The results of the Simplex optimisation process developed in this work are shown in Table 5.6(a) for the starting 'seed' values of M_p=16.01 and r=0.51. The GA optimised results and their use as starting 'seed' values in the hybrid optimisation of the Simplex method are shown in Tables 5.6(b) and (c) respectively. Again, as in design example 2, the objective function profile indicates regions of sufficient flatness to prevent convergence of the Simplex method towards a global minima point. The starting 'seed' values for the Simplex method as obtained using the GA optimised results are again useful in converging towards improved results for this design example.

The best optimised results taken from Table 5.6(c) generate coefficients of the prototype filter as shown in Table 5.7. Note that only the first half of the coefficients are shown due to the linear phase characteristic of the prototype filter. These values were used to plot the frequency responses of the prototype filter and the eight analysis filters of the pseudo-QMF bank as shown in Figures 5.14 (a) and (b) respectively. The linear and aliasing distortion responses are shown in Figures 5.15 (a) and (b) respectively. The Simulink test error signal for a random input with uniform distribution in the range −1 to 1 is shown in Figure 5.16. Significant improvement of these optimised results is evident when compared to the estimated results given by Fliege [1994] and shown in Table 5.6(d).

Table 5.6

(a) Optimised results using downhill Simplex method with seed values M_p=16.01 and r=0.51 for N=257

α	M_p	r	c	E_{pp}	E_A	v_{rms}	out_{pp}
0.0100	15.9966	0.5289	0.9997	0.0062	0.0001	0.0008	0.0052
0.1000	16.0021	0.5030	1.0001	0.0016	0.0002	0.0003	0.0036
0.2000	16.0022	0.5029	1.0001	0.0016	0.0002	0.0003	0.0036
0.3000	16.0021	0.5028	1.0001	0.0016	0.0002	0.0003	0.0036
0.4000	16.0022	0.5030	1.0001	0.0016	0.0002	0.0003	0.0036
0.5000	16.0021	0.5022	1.0001	0.0016	0.0002	0.0003	0.0035
0.6000	16.0021	0.5022	1.0001	0.0016	0.0002	0.0003	0.0035
0.7000	16.0021	0.5022	1.0001	0.0016	0.0002	0.0003	0.0035
0.8000	16.0021	0.5017	1.0001	0.0016	0.0002	0.0004	0.0035
0.9000	16.0021	0.5017	1.0001	0.0016	0.0002	0.0004	0.0035
1.0000	16.0021	0.5017	1.0001	0.0016	0.0002	0.0004	0.0035

(b) Optimised results using GA with M_p range=13 to 19, 'r' range = 0 to 2 and alpha range= 0.1 to 0.9 for N=257.

α	M_p	r	c	E_{pp}	E_A	v_{rms}	out_{pp}
0.1045	16.0169	1.0095	1.0005	0.0020	0.0001	0.0003	0.0027

(c) Optimised results using hybrid downhill Simplex method with seed values taken from GA results of Table 5.3 (c) for N=257.

α	M_p	r	c	E_{pp}	E_A	v_{rms}	out_{pp}
0.1045	15.9989	1.0037	1.0000	0.0008	0.0001	0.0002	0.0022

(d) Estimated results taken from graphical plots of Fliege [1994] for N=257.

E_{pp}	E_A
0.0085	0.0008

Table 5.7

Optimised prototype filter coefficients using parameters taken from Table 5.6(c) for design example 3 (N=257). Only the first half of the coefficients are shown.

n	p(n)	n	p(n)	n	p(n)	n	p(n)
0	-7.6911E-05	33	-1.3319E-04	66	-2.4403E-04	99	-6.1474E-04
1	-7.5048E-05	34	-1.0822E-04	67	-1.4442E-04	100	-3.0059E-05
2	-6.1428E-05	35	-6.5170E-05	68	-1.4971E-05	101	6.5227E-04
3	-3.7787E-05	36	-9.9664E-06	69	1.2598E-04	102	1.3374E-03
4	-7.4663E-06	37	4.9322E-05	70	2.5703E-04	103	1.9146E-03
5	2.5047E-05	38	1.0363E-04	71	3.5682E-04	104	2.2708E-03
6	5.4780E-05	39	1.4426E-04	72	4.0736E-04	105	2.3071E-03
7	7.7025E-05	40	1.6425E-04	73	3.9702E-04	106	1.9554E-03
8	8.8079E-05	41	1.5952E-04	74	3.2287E-04	107	1.1940E-03
9	8.5848E-05	42	1.2965E-04	75	1.9172E-04	108	5.9326E-05
10	7.0226E-05	43	7.8110E-05	76	1.9851E-05	109	-1.3471E-03
11	4.3179E-05	44	1.1918E-05	77	-1.6880E-04	110	-2.8593E-03
12	8.5121E-06	45	-5.9296E-05	78	-3.4573E-04	111	-4.2542E-03
13	-2.8655E-05	46	-1.2465E-04	79	-4.8195E-04	112	-5.2669E-03
14	-6.2635E-05	47	-1.7363E-04	80	-5.5240E-04	113	-5.6123E-03
15	-8.8034E-05	48	-1.9775E-04	81	-5.4018E-04	114	-5.0115E-03
16	-1.0060E-04	49	-1.9197E-04	82	-4.3988E-04	115	-3.2210E-03
17	-9.7902E-05	50	-1.5563E-04	83	-2.5942E-04	116	-6.0860E-05
18	-7.9811E-05	51	-9.2826E-05	84	-1.9987E-05	117	4.5601E-03
19	-4.8568E-05	52	-1.1967E-05	85	2.4596E-04	118	1.0626E-02
20	-8.5377E-06	53	7.5256E-05	86	4.9869E-04	119	1.8002E-02
21	3.4385E-05	54	1.5553E-04	87	6.9666E-04	120	2.6436E-02
22	7.3634E-05	55	2.1590E-04	88	8.0269E-04	121	3.5571E-02
23	1.0296E-04	56	2.4579E-04	89	7.9019E-04	122	4.4963E-02
24	1.1743E-04	57	2.3877E-04	90	6.4835E-04	123	5.4113E-02
25	1.1420E-04	58	1.9382E-04	91	3.8546E-04	124	6.2502E-02
26	9.3071E-05	59	1.1577E-04	92	2.9746E-05	125	6.9635E-02
27	5.6630E-05	60	1.4895E-05	93	-3.7273E-04	126	7.5076E-02
28	9.9320E-06	61	-9.4338E-05	94	-7.6307E-04	127	7.8485E-02
29	-4.0169E-05	62	-1.9529E-04	95	-1.0770E-03	128	7.9646E-02
30	-8.6007E-05	63	-2.7157E-04	96	-1.2542E-03		
31	-1.2026E-04	64	-3.0963E-04	97	-1.2477E-03		
32	-1.3711E-04	65	-3.0101E-04	98	-1.0330E-03		

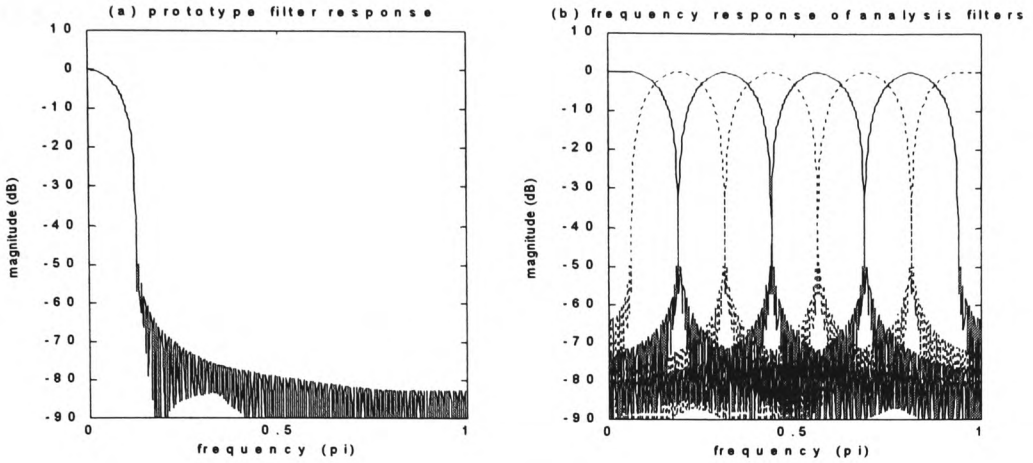

Figure 5.14 Magnitude frequency response for N=257 (Table 5.6(c) optimised results): (a) prototype filter and (b) all channels.

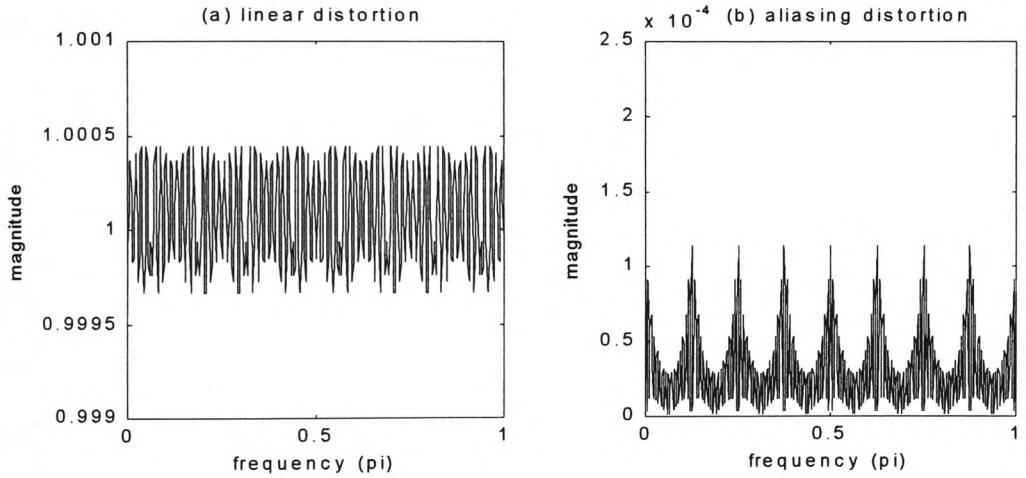

Figure 5.15 Overall distortion for N=257 (Table 5.6(c) optimised results): (a) linear and (b) aliasing.

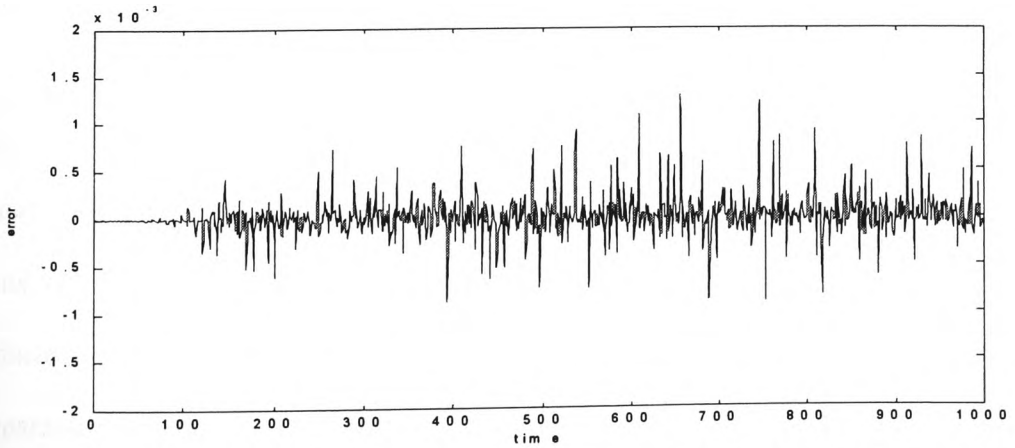

Figure 5.16 Simulink test error signal for a random input with a uniform distribution [-1,1] for N=257 (Table 5.6(c) optimised results).

5.5 Discussion of results

The design of a class of uniform M-channel filter bank considered in this chapter is based on the cosine modulation technique. A single low pass prototype FIR filter is used to derive the M-channel filters that are all of equal width and are equally spaced on the frequency scale. The approximate closed form expressions of the impulse response of the analysis and synthesis filters are derived such that only the directly adjacent aliasing spectra is compensated and reduced. However, there is still a problem with the amplitude distortion that must be considered for minimising the overall error. It is this limitation and an opportunity for reducing the aliasing and the magnitude distortions that was inspirational towards the study covered in this chapter.

The design of the low pass prototype filter is based on a square root raised cosine characteristic [Fliege, 1994]. This form of filter offers flexibility in terms of the cut-off frequency and roll-off factor that are found to be useful parameters in the optimal design of the uniform filter bank. It must be recognised that although the low pass prototype filter is of a linear phase FIR form, the analysis and synthesis filters derived from the prototype are FIR but in general, have a non-linear phase response. The closed form expressions for the uniform filter bank design eliminates phase distortion and is therefore, not relevant for consideration in the overall optimisation scheme.

Tests were conducted for three design examples all using M=8 channels but with different number of coefficients of the FIR prototype filter. This choice was based on the available results reported in literature by other researchers so that a comparative study could be conducted. In design example 1, the number of coefficients used is N=39. This design is comparable with the results reported in [Vaidyanathan, 1993]. The new GA and hybrid optimised results for this design example show a substantial reduction of the overall distortion

when compared with the optimised results generated using a non-linear optimisation package as reported in [Vaidyanathan, 1993]. The complete results are shown in Tables 5.1 and 5.2 and the graphical results are plotted in Figures 5.5 to 5.10.

In design example 2, the number of coefficients used is N=141. Although no direct comparison of this design example is available in literature, it is instructive to compare the new optimised results with the graphical results of Fliege [1994] for which N=257. The new optimised results as shown in Table 5.4 for N=141 are a significant improvement over the graphical results of Fliege [1994] as seen in Table 5.6 (d) for N=257. Furthermore, the new optimised results of design example 3 for N=257 show a small improvement over the results of design example 2. This is instructive as it clearly leads to the choice that a design engineer has in terms of quality of the system and the computational overheads.

5.6 Summary and conclusions of Chapter 5

A novel approach to the minimisation of the amplitude distortion and aliasing error for a maximally decimated M-channel uniform filter bank has been investigated in this chapter. The use of a square root raised cosine prototype filter to obtain an M-channel uniform filter bank by cosine modulation technique has been considered. The closed form solution for the impulse response of the analysis and synthesis FIR filters eliminates phase distortion and minimises aliasing error. It remains then to minimise the amplitude distortion by some optimisation process. One example of an 8-channel uniform filter bank given by Vaidyanathan [1993, pp. 336] uses a prototype filter for which the coefficients were optimised using a non-linear optimisation package [Press et al, 1989]. This is based on minimisation of a composite objective function formed using a trade-off parameter between the amplitude distortion and the stop band attenuation of the prototype filter. The emphasis is towards minimising the amplitude

distortion with the implicit acceptance that aliasing error is sufficiently small and therefore, not significant in the overall performance of the filter bank system.

The novel contribution towards the optimisation process as covered in this chapter, is to use the flexibility of the square-root raised cosine FIR prototype filter for which the bandwidth and the roll-off factor are allowed to be perturbed in the quest for optimality. The objective function that must be minimised is of a composite form using a trade-off parameter between the maximum peak-to-peak amplitude distortion E_{pp} and the maximum overall aliasing error E_A. This procedure does not preclude the aliasing error from being used as a parameter that may influence the overall quality of the system. Furthermore, there is an opportunity to study the effect of amplitude distortion and aliasing error and their inter-dependence on the overall performance of the system.

While the raw minimisation results for E_{pp} and E_A in themselves give only a brief understanding of the system performance, a better understanding is achieved by conducting the Matlab Simulink toolbox tests of the full system as shown in Figure 5.2. The error signal generated from these tests give a predictable measure of the system behaviour under real-time conditions. A plot of E_{pp} and E_A against the trade-off parameter α taken from the results of the design example 1, Table 5.1(a), is shown in Figure 5.17. This plot also includes the corresponding root mean square value v_{rms} of the error signal taken from the Simulink tests of Figure 5.2. A clear dependence of the maximum aliasing error E_A is evident on the overall performance of the system with v_{rms} reducing with decreased E_A.

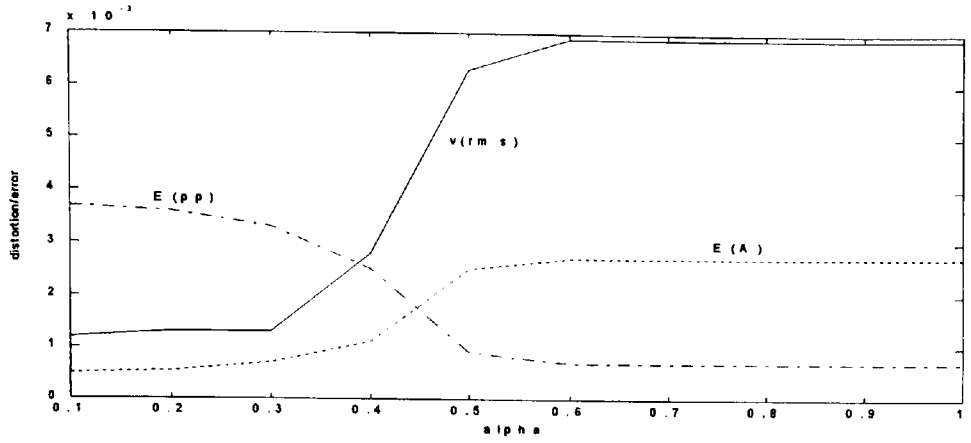

Figure 5.17 A plot of maximum peak to peak magnitude distortion E_{pp}, maximum aliasing error E_A and root mean square v_{rms} value of the Simulink error signal against the trade-off parameter α for test results of design example 1, Table 5.1(a).

Three optimisation methods were used to minimise the composite objective function. These are; the non-gradient based Simplex method, the gradient based unconstrained quasi-Newton method and the genetic algorithm method. These were applied to the case of three design examples based on different number of coefficients of the prototype filter i.e. N=39, 141 and 257. The general outcome of the results showed a dependency of the starting 'seed' values on the optimally converged values for both the Simplex and the quasi-Newton methods. The genetic algorithm method generated good results although further minimisation was possible with a hybrid procedure of applying the Simplex method by using the GA optimised results as the starting 'seed' values. A further enhancement of the GA code was the inclusion of the trade-off parameter α as the third variable that assisted in automating the optimisation procedure.

A comparison of the results for the design example 1 shows a significant improvement over the results of Vaidyanathan [1993]. This is clearly evident when comparing both the amplitude distortion and the aliasing error as shown in Figures 5.6 and 5.9. Further tests using the Simulink toolbox show a substantial reduction of the error signal for the new optimisation procedure as seen in Figure 5.7 when compared to the reconstructed results of Vaidyanathan [1993] as seen in Figure 5.10. The results of design example 3 i.e. for N=257 can be compared

with the graphical results of Fliege [1994]. Again significant improvement of the new results is evident. The design example 2 using N=141 has no other source for comparison. However, it is instructive to compare these results with those of design example 3 for N=257. The improvement of amplitude distortion, aliasing error and the Simulink error signal is fairly small for a substantially larger number of coefficients used for the filter bank design in example 3. Thus there is clearly a choice for the designer in terms of computational overheads against the overall performance of the filter bank system.

The major contributions of this part of the study are the following.

- A real-valued genetic algorithm code has been developed for the optimisation of a uniform maximally decimated M-channel filter bank. Further modification of this code involved inclusion of the quasi-Newton and the Simplex codes for a comparative and hybrid study.
- The design of the uniform M-channel filter bank is based on the cosine modulation technique, for which a square root raised cosine form of a prototype filter is used. The design optimisation process involved perturbing the bandwidth and the roll-off factor of the prototype filter. Significantly improved new results using the GA and hybrid optimisation are given in Tables 5.1, 5.4 and 5.6.

The new results reported in this chapter shows an effective means for realising an optimal uniform multirate filter bank that is based on FIR filters. The design is direct and is easily implemented in real time. For real signals, however, the ensemble average of the energy varies in the frequency sub-bands that may be of unequal widths. Thus for optimal coding efficiencies, non-uniform filter banks are desirable. This issue will form the basis of the study, investigation and optimisation that is covered in the next chapter.

Chapter 6: Optimisation of a class of M-channel non-uniform filter bank

Overview of Chapter 6: This chapter deals with the issues of non-uniform multirate filter banks. This form of filter bank can split the input signal into non-uniformly spaced segments on the frequency scale thus leading towards efficient coding gains of real time signals. Two design methods are considered here; the first method is based on the transformation of a FIR prototype filter by means of sine or cosine multiplication to the designated frequency bands. The second method is based on using multiple square root raised cosine prototype filters from which the analysis and synthesis filters are derived using the closed form approximation as covered in Chapter 5. Some theoretical issues are included and the optimisation of the filter bank design using GAs and a hybrid approach is investigated. Finally, several design examples are tested and the results reported.

6.1 Introduction

The issues of design and optimisation considered in Chapter 2 for finite word length FIR digital filters and for multirate filter banks considered in Chapters 4 and 5 are extended to the case of non-uniform multiple-band filter banks that is studied in this Chapter. The prototype filters used for the design of the non-uniform filter banks considered here are based on the use of FIR analysis and synthesis filters. For real signals, the ensemble average of the energy can vary significantly in different frequency bands. High coding gain is thus achievable by using non-uniform multirate filter banks (NUF) as shown in Figure 6.1.

All filter banks suffer from the usual problems of amplitude, phase and aliasing errors thus the constraints for perfect reconstruction (PR) can be extensive. The case for maximally decimated

M-channel uniform filter banks, as covered in Chapter 5, is well studied and PR conditions are well established [Vaidyanathan, 1993]. Non-uniform banks, however, have particular problems for PR for which extensive constraints exist. Mostly, therefore, the problem is reduced to relaxing constraints at the expense of errors and finding methods for minimising the errors. Consequently, optimisation techniques are commonly used in the design, development and implementation of non-uniform filter banks. Several examples of NUF bank design and optimisation techniques have been reported in literature. Hoang and Vaidyanathan [1989] cover the concept of a compatible set 'r_k' and its requirement for cancelling aliasing. Design methods that allow perfect reconstruction and transformation of non-uniform into uniform filter banks is reported in [Kovacevic and Vetterli, 1993]. Other examples of NUF bank design based on the use of multiple-prototype LP filters and cosine modulation that allows cancellation of the main aliasing component are considered in [Wada, 1995], [Jeong-Jin and Byeong, 1995], [Argenti and Del Re, 1996] and [Argenti et al, 1998]. Mehr and Chen [1999, 2000] have reported the optimal design of NUF banks by minimisation of the H-2 and H-infinity norms of the error system. Another approach based on the time-domain design of FIR filters is reported by Nayebi et al [1993].

In this Chapter, two methods based on direct design approach using FIR low pass prototype (baseband) filters are considered. The first design method uses transformation of the prototype filters by sine or cosine multiplication as developed by Chu [1985] and Wada [1995]. The second method is an extension of the design procedure covered in Chapter 5 of this thesis that uses multiple square root raised cosine prototype filters from which the analysis and synthesis filters are derived using the closed form approximation. The optimisation process considered in the present work applies to the overall NUF bank as shown in Figure 6.1, including the decimators and expanders. This method leads to the minimisation of amplitude and aliasing distortion in a combined manner. An impulse (in the form of a dirac-delta function) is applied

to the network of Figure 6.1 and the Fourier transform of the response is calculated, giving the transfer function of the network. In the case of the first design method, the cut-off frequencies of the prototype filters are allowed to vary independently by a small amount δf_{ck} thus changing the bandwidth of the analysis and synthesis filters by a small amount. For the second design method the bandwidth parameter 'M_p' and the roll-off factor 'r' of individual prototype filters are allowed to change by a small amount thus leading towards an optimised system. The optimisation process is applied to the transfer function of the network to minimise the distortion. For this optimisation procedure, it was observed that a direct use of gradient based quasi-Newton or down hill Simplex methods are highly sensitive to starting 'seed' values. There is no assurance of a global minima point being attained. For this reason, the genetic algorithm method is used to search for good results over a wider landscape of frequency response variations for each set of prototype filters. The standard minimisation methods are then applied to further optimise the results through a hybrid approach.

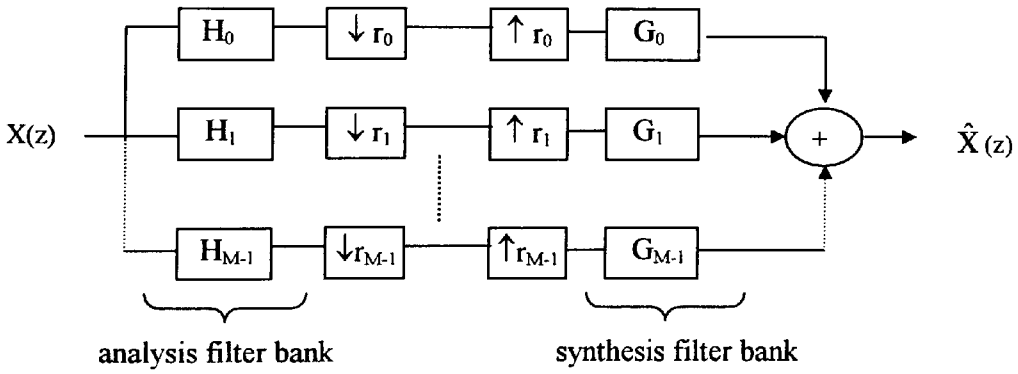

Figure 6.1 Non-uniform multirate filter bank with integer decimators and integer expanders.

6.2 Theory and design issues

An M-channel non-uniform filter bank with integer decimation factors r_k, k = 0, ..., M-1 as shown in Figure 6.1 will be considered here. The input-output relationship in the z-domain is given by:

$$Y(z)=\sum_{k=0}^{M-1} G_k(z)\frac{1}{r_k}\sum_{\ell=0}^{r_k-1} H_k(z W_{r_k}^{\ell}).X(z W_{r_k}^{\ell}) \qquad 6.1$$

where $W_{r_k} = e^{-j2\pi/r_k}$

The design of the NUF bank will be constrained to the case of maximally decimated integer values for which the following applies

$$\sum_{k=0}^{M-1}\frac{1}{r_k} = 1 \qquad 6.2$$

Most examples of NUF banks fall into the category of 'incompatible sets' of r_k i.e. at least one shifted copy of $X(e^{j\omega})$ does not have a compatible pair at the output of another expander. In this case, complete elimination of aliasing is impossible for non-ideal filters. However, such filter banks may be resolved into a tree-structured form representing uniform filter banks [Hoang and Vaidyanathan, 1989]. This property is useful in working out an overall aliasing error [Argenti and Del Re, 1996]. Assuming L is the least common multiple of the r_k's and R_k are integers such that $L= r_k R_k$. Then, the non-uniform bank is resolved into an L-channel uniform bank with the new transfer functions of analysis filters given by: $z^{tr_k}H_k(z)$, t=0,...., R_k-1 and $z^{-tr_k}G_k(z)$, t=0,...., R_k-1 for synthesis filters. The reconstructed output signal is then given by

$$\hat{X}(z)=\sum_{\ell=0}^{L-1} X(z W_L^{\ell})\ A_\ell(z) \qquad 6.3$$

where $A_\ell(z)$ denotes the components given by

$$A_\ell(z) = \frac{1}{L}\sum_{k=0}^{M-1} H_k(z W_L^{\ell})\, G_k(z) \sum_{t=0}^{R_k-1} W_L^{\ell t r_k} \qquad 6.4$$

For $\ell=0$, the linear distortion function is obtained and is given by

$$A_0(z) = \frac{1}{r_k}\sum_{k=0}^{M-1} H_k(z)\, G_k(z) \qquad 6.5$$

In the absence of aliasing, $A_0(z)$ is the transfer function and the NUF bank is a linear time invariant system. The various aliasing components $X(z W_L^{\ell})$ of the input signal, where

ℓ=1,....L-1, that do not cancel each other out should therefore, be considered separately. An overall aliasing error function is defined as

$$A_{alias}(z) = \left[\sum_{\ell=1}^{L-1} \left| A_{\ell}(z) \right|^2 \right]^{1/2} \qquad 6.6$$

6.2.1 Design method one

For the case of design method one, the bandwidth of the analysis filter for the positive frequency range 0 to + π is given by

$$BW_k = \frac{\pi}{r_k} \quad (k = 0,\ldots., M-1) \qquad 6.7$$

The centre frequency of the analysis filters from 0 to π is given by

$$\omega_{c_{k+1}} = \omega_{c_k} + \frac{\pi}{2}\left(\frac{1}{r_k} + \frac{1}{r_{k+1}}\right) \quad (k = 1,\ldots, M-3) \qquad 6.8$$

where $\omega_{c_0} = 0$, $\omega_{c_1} = \pi\left(\frac{1}{r_0} + \frac{1}{2r_1}\right)$ and $\omega_{c_{M-1}} = \pi$

The required transfer functions of the analysis filters are given by

$$H_k(z) = \begin{cases} 1 & \omega_{c_k} - \frac{\pi}{2r_k} < |\omega| < \omega_{c_k} + \frac{\pi}{2r_k} \\ 0 & \text{elsewhere} \end{cases}$$

for k = 1,....., M - 2 $\qquad$ 6.9

also

$$H_0(z) = \begin{cases} 1 & |\omega| < \frac{\pi}{r_0} \\ 0 & \text{elsewhere} \end{cases} \qquad 6.10$$

and

$$H_{M-1}(z) = \begin{cases} 1 & \pi - \frac{\pi}{r_{M-1}} < |\omega| < \pi \\ 0 & \text{elsewhere} \end{cases} \qquad 6.11$$

The design of the analysis filters is based upon coefficient optimised low pass FIR prototype filters of the impulse response $v_k(n)$. The impulse response of analysis filters is then obtained using

$$h_k(n) = \begin{cases} v_k(n).\cos\{\omega_{c_k}(n - \frac{N+1}{2})\} & \text{for k: even} \\ v_k(n).\sin\{\omega_{c_k}(n - \frac{N+1}{2})\} & \text{for k: odd} \end{cases} \qquad 6.12$$

Where the number of coefficients N=MK and K is an odd number. Also, n=1,...,N and k=0,...,M-1. ω_{c_k} represents the centre frequency of the analysis filters and the impulse response then becomes $2h_k(n)$ for k=1,....M-2.

The synthesis filters are obtained using

$$G_k(z) = (-1)^k H_k(z) \qquad 6.13$$

for k = 0,..., M – 1

The impulse response of prototype LP filters $v_k(n)$, is designed with cut-off frequencies given by π/r_k for k=0 and k=M-1 and $\pi/2r_k$ for k=1,...,M-2. These cut-off frequencies are independently perturbed slightly to obtain an optimised transfer function of the complete network of Figure 6.1.

6.2.2 Design method two

This method is based on the design of a cosine-modulated pseudo-QMF filter bank for which the constraint is to obtain an appropriate transfer function of the prototype filter that will generate approximately power complementary frequency responses of frequency-shifted replicas about the centre frequencies. A commonly used prototype filter in this application of the filter bank is approximated by the square root raised-cosine characteristic [Fliege, 1994].

The frequency and impulse response for such a low pass filter is given by Equation 5.11 and 5.12 respectively. The analysis and synthesis filters are derived using Equations 5.15 and 5.16 respectively.

The non-uniform filter bank is derived from the pseudo-QMF cosine modulated uniform filter bank by using multiple prototypes p(n) and deriving the appropriate analysis and synthesis filters represented by Equations 5.15 and 5.16. The optimisation process is then based upon using the prototype variables i.e. bandwidth $\pi/2M = \pi/M_p$ and the roll-off factor 'r' for each of the low pass FIR prototype filters to optimise the overall distortion of the filter bank.

6.2.3 Design examples and tests

A number of design examples are considered here based on the two design methods mentioned above. Two broad categories of design examples covered are based on a 3-band and a 5-band non-uniform filter banks. Both of these filter banks conform to the case of maximally decimated integer valued structures as shown in Figure 6.1. For each of these NUF banks, tests are conducted using different lengths of FIR filter prototypes. A comparison is then drawn between each of the test results based on the optimised results for the following errors.

i) The maximum peak to peak ripple of the amplitude distortion E_{pp} given by

$$E_{pp} = \max[|A_0(z)|] - \min[|A_0(z)|] \qquad 6.14$$

ii) Aliasing distortion that is derived by taking the maximum value of $A_{alias}(z)$ over all ω. This gives the worst possible peak aliasing distortion i.e.

$$E_A = \max[A_{alias}(z)] \qquad 6.15$$

iii) Matlab Simulink tests based on a test circuit of the form shown in Figure 6.2 using a random signal at the input with uniform distribution in the range −1 to +1. The error signal v_e is then obtained from the difference between the output signal and an

appropriately delayed version of the input signal. This error signal is used to calculate the root mean square value given by v_{rms} and the maximum peak to peak error voltage given by out_{pp}.

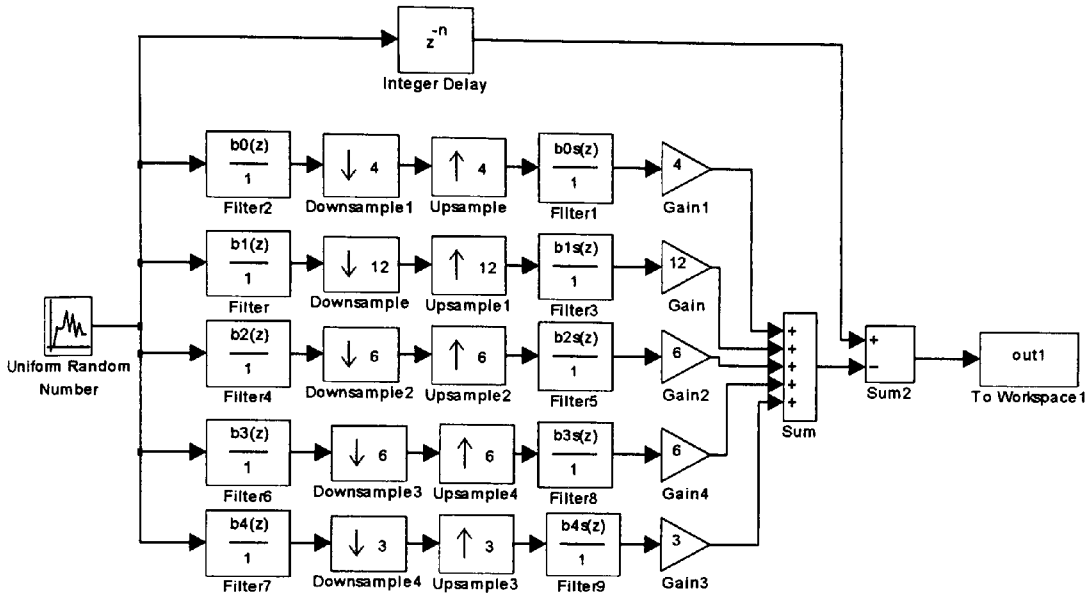

Figure 6.2 A Matlab Simulink test circuit for a 5-band non-uniform filter bank.

6.2.4 Optimisation methods

Three optimisation methods are used in this section of the study. The first method is based on using the downhill Simplex algorithm due to Nelder and Mead [1965] and has been implemented as the `fmins.m` function of the optimisation toolbox of Matlab. The second method is a gradient based unconstrained quasi-Newton method implemented as `fminu.m` function of Matlab. The third method is based on a simple real-valued genetic algorithm developed as a Matlab toolbox by Chipperfield et al [1993]. Two GA codes were developed for the two design methods applied for this work. These codes are shown in Appendices F1.1 and F1.2 respectively. The choice for the selection of these three methods of optimisation is based the same reasoning as mentioned in section 5.3.5 of Chapter 5. The results of these optimisation

methods have again shown that the hybrid approach of generating good 'seed' values using the GA method and then applying the Simplex or the quasi-Newton method results in good minimisation of the objective function.

6.2.5 GA optimisation methodology and pseudo code

The genetic algorithm used here for the design of the non-uniform M-channel filter bank is identical to the generic form explained in section 1.4, Chapter 1. This is a Matlab based algorithm developed originally for control systems applications [Chipperfield et al, 1993]. The main GA code has been adapted for the application in this work and new functions have been written for working out the error objective function. The specific steps followed for the design stage of the QMF bank GA optimisation are shown here and the pseudo GA code is shown in Figure 6.3 and the pseudo code to evaluate the objective function is shown in Figure 6.4.

1) Define the GA parameters

The GA parameters used for the various design examples are given by

```
GGAP=0.8;       % generational gap
INSR=0.8;       % reinsertion rate
MAXGEN=10;      % number of generations
Nind=100;       % population size
MutRate=0.01;   % mutation rate
```

2) Create population set of individuals

The starting set of parameter values for design method 1 is based on the variation of the cut-off frequency δf_{c_k} of each low pass prototype filter used. For design method 2, the variations of the bandwidth M_p and the roll-off factor r for the low pass prototype filters are used. The bounded parameter values are described in a matrix 'FieldDR' and an initial population set consisting of

random real-valued individuals is created within the bounds specified in FieldDR matrix. The function **crtrp** of the GA Matlab toolbox is used for this purpose.

3) The Objective function evaluation

The main purpose of the optimisation process here is to minimise the objective function with the specific aim of minimising the overall magnitude and aliasing error so that a perfect reconstruction characteristic is closely met. The error objective function to be minimised and used in the optimisation process is given by

$$E = \max [20 \log_{10}|FFT(y(n))|] - \min [20 \log_{10}|FFT(y(n))|] \qquad 6.16$$

where y(n) is the impulse response of the network of Figure 6.1 when the input impulse applied is given by x(n)=1,0,0.....0.

4) Fitness value and ranking

The Matlab based **ranking** function of the GA toolbox ranks the individuals according to their objective function values 'Obj_err' and returns a column vector consisting of the corresponding fitness value 'FitnV' of the individuals. This function performs a linear ranking with a selective pressure (SP) of 2 [Whitley, 1998]. The fitness value assigned to the individuals is calculated according to the formula given by Equation 1.1 in Chapter 1.

5) Selection of individuals for breeding

The high-level function for selection of individuals from the population set and returning the selected individuals in a new population is performed by the **select** function. The low-level selection function **sus** is called by the **select** function. The **sus** function is based on a form of stochastic sampling method and is implemented by obtaining a cumulative sum of the fitness vector 'FitnV' and generating a set of equally spaced numbers between 0 and Σ(FitnV) [Baker, 1987]. The probability of an individual being selected is given by Equation 1.2 in Chapter 1.

6) Recombining individuals – crossover

The crossover function is also performed in two stages. The high-level function is **recombin** that calls the low-level function **recdis**. The **recdis** function is a discrete recombination function. The mating process is performed between pairs of rows. The **recdis** function first generates an internal mask table that determines which parents contribute which variables to the offspring. On the basis of the randomly generated mask table, the variable values are exchanged between the individuals and return a new population after mating.

7) Mutation

The **mutbga** function of the Matlab GA toolbox takes real-valued population, mutates each variable with given probability and returns the population after mutation. The **mutbga** function produces firstly a random internal mask table that determines which variables will mutate and also the sign for the step size. A second internal table generates the normalised mutation step size. The mutated variable is worked out as a function of the original variable and the step size [Muhlenbein and Schlierkamp-Voosen, 1993].

8) Reinsert offspring into new population

The new population set generated after crossover is subjected to the objective function evaluation of each new individual. On the basis of their fitness, the offspring are selected for reinsertion using the **reins** function into the new population. The objective function values are then copied to the reinserted offspring and the GA loop is then repeated for the next generation.

```
% pseudo GA and hybrid optimisation code for non-uniform filter bank

% GA characteristics here

% low pass prototype filter variable parameters

% build a field descriptor for search space
FieldDR = [-a -a -a -a -a ;    % lower bound
           a a a a a];         % upper bound
% create initial population
Chrom = crtrp(Nind, FieldDR);
%  generational counter

while gen < MAXGEN

   FitnV = ranking(ObjVal);                    % fitness

   SelCh = select('sus',Chrom,FitnV,GGAP);     % selection

   SelCh = recombin('recdis', SelCh, 1);       % recombine - crossover

   SelCh = mutbga(SelCh,FieldDR, MutRate);     % mutation

   ObjVOff = feval('nuf5b_obj',SelCh);         % evaluate objective fn.

[Chrom, ObjVal] = reins(Chrom, SelCh,1,1, ObjVal, ObjVOff); % reinsert

   gen = gen + 1

xbest = Chrom(ix,:);
end

x(1)=xbest(1);x(2)=xbest(2);x(3)=xbest(3);x(4)=xbest(4);x(5)=xbest(5);


% second stage for optimisation using Simplex algorithm of the form
% x=fmins('fun5b_obj',x)

% new optimised parameters of the prototype filters
```

Figure 6.3 Pseudo GA and hybrid optimisation code for the non-uniform filter bank.

```
% pseudo objective function code
function f = nuf5b_obj(Chrom);
% starting values

% evaluate overall transfer function
yt=20*(log10(abs(fft(bt))));

Obj_err = max(yt(1:5000))-min(yt(1:5000));
end
```

Figure 6.4 Pseudo objective function code for the non-uniform filter bank GA code.

In this work, genetic algorithms have been used to generate a pool of population set of small independent perturbations of the prototype LP filter cut-off frequencies given by δf_{ck}, k=0,.....,M-1 for the case of design method 1. The procedure of ranking, crossover and mutation are then applied through a number of generations to obtain a set of variables that minimise the given objective function. This allows for search over a wide landscape of possible solutions. Further optimisation is then applied using the standard quasi-Newton or the down-hill Simplex algorithm. A block diagram of the complete process is shown in Figure 6.5. A similar optimisation procedure is applied for the case of design method 2 where the variables are bandwidth and roll-off parameters.

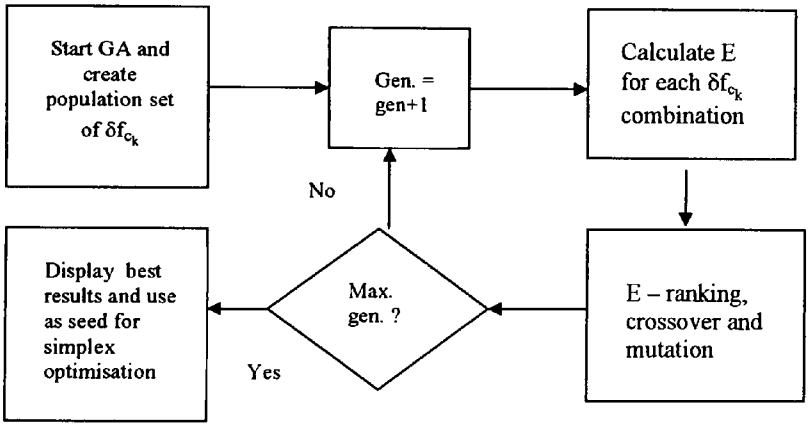

Figure 6.5 Flow chart showing the GA optimisation procedure.

6.3 Some results

A number of tests were conducted based on the two design methods mentioned above and using different number of non-uniform bands and coefficients for the low pass filter prototypes. Design method 1 is based on the direct transformation of low pass prototypes developed using `fir1.m` function of Matlab. This function generates the classical Hamming-windowed, linear phase filter with cut-off frequency f_{ck}. Transformation of the prototype filters is achieved by

sine or cosine multiplication to the appropriate band as developed by Chu [1985] and Wada [1995]. Some comparison is also made with the graphical results of Wada [1995], although no reconstruction was possible since the coefficient values for the prototype filters are not specified. However, the optimisation procedures developed in the present work were applied to this example using the same number of coefficients (i.e. N=45 for a 5-band non-uniform filter bank) and some comparisons were made based on the visually observable results of Wada [1995].

Design method 2 is based on an extension of the method used in Chapter 5 for the development of uniform filter banks. Multiple prototype low pass filters developed using the square root raised cosine method are used for the case of non-uniform filter banks. The method of pseudo-QMF bank using cosine modulation is then applied to derive the analysis and synthesis filters using the closed form approximation. The variables in this case are the bandwidth parameter M_p and the roll-off factor 'r'.

Three design examples are considered here and all of these fall in the category of maximally decimated NUF banks. These are:

Example 1: A 3-band structure with r_0=2, r_1=6, r_2=3 and N=27 and 51.

Example 2: A 5-band structure with r_0=4, r_1=12, r_2=6, r_3=6, r_4=3 and N=45 and 65 – case 1.

Example 3: A 5-band structure with r_0=8, r_1=8, r_2=4, r_3=4, r_4=4 and N=31, 45 and 65 – case 2.

All of the above design examples fall in the category of 'incompatible sets' i.e. at least one shifted copy of $X(e^{j\omega})$ does not have a compatible pair at the output of another expander. In this case, complete elimination of aliasing is impossible for non-ideal filters. However, these examples can be resolved into a tree-structured form representing the uniform filter banks

[Hoang and Vaidyanathan, 1989]. This property is useful in working out the overall aliasing error function given by Equation 6.6.

6.3.1 Design example 1: For a 3-band NUF bank

Design method 1

The 3-band non-uniform filter bank structure is designed using r_k values of: $r_0=2$, $r_1=6$ and $r_2=3$ for which M=3. The results for the design method 1 for N=27 and 51 are shown in Figures 6.6 and 6.8 respectively. The corresponding Simulink test error signal results are shown in Figures 6.7 and 6.9 respectively. The tabulated results for the 3-band NUF bank are shown in Table 6.1 where E_{pp} is the maximum peak to peak amplitude distortion as given by Equation 6.14 and E_A is the maximum aliasing distortion given by Equation 6.15. δf_{c0}, δf_{c1} and δf_{c2} are small frequency perturbations of the cut-off frequency of the prototype filters. Also, v_{rms} and out_{pp} are the root mean square and maximum peak to peak values for the Simulink test results of the error signal.

Table 6.1 Results for the 3-band NUF bank using design method 1.

N	E_{pp}	E_A	δf_{c0}	δf_{c1}	δf_{c2}	v_{rms}	out_{pp}
27	0.0809	0.5436	0.0338	0.0276	0.0290	0.0145	0.1103
51	0.0143	0.3765	0.0162	0.0161	0.0162	0.0024	0.0156

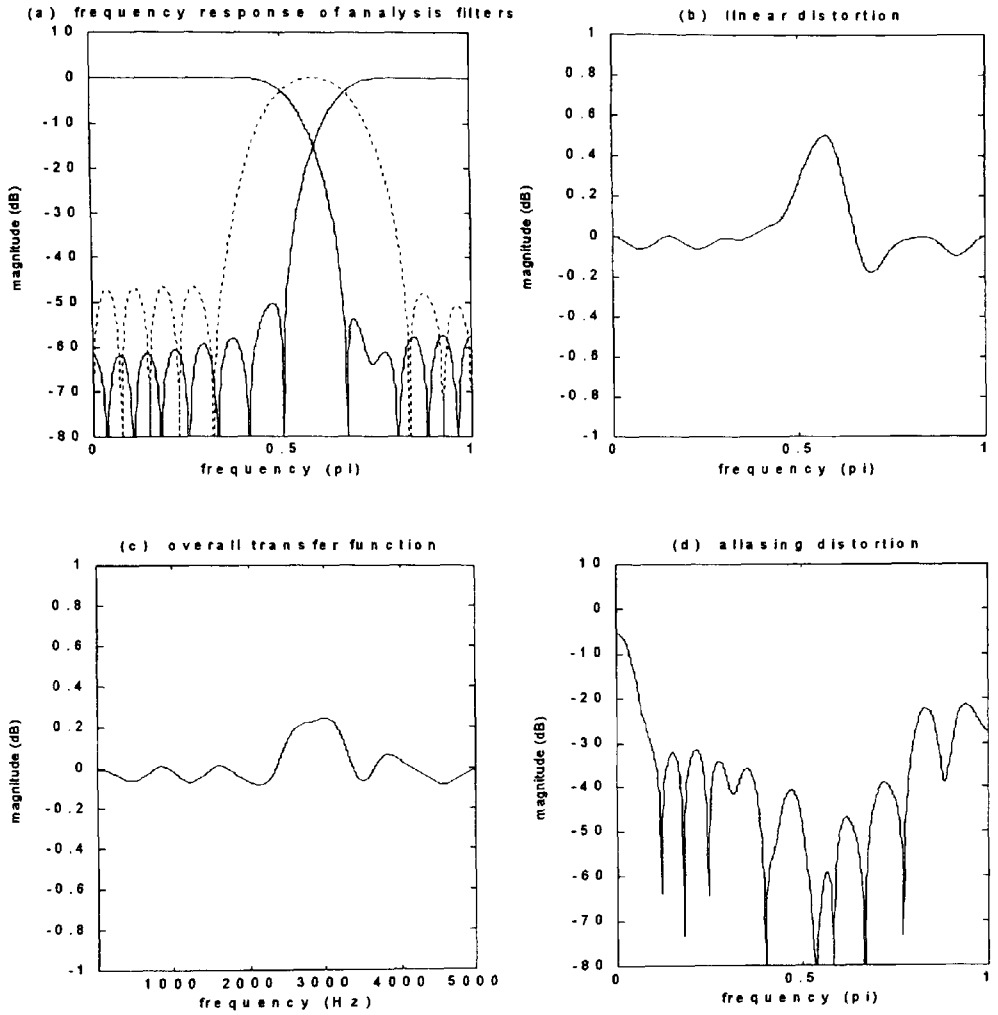

Figure 6.6 For a 3-band NUF bank using design method 1 and N=27 (a) magnitude frequency response of all analysis filters (b) transfer function of the network assuming aliasing is zero (c) overall transfer function of the network of Figure 6.1 (d) aliasing distortion.

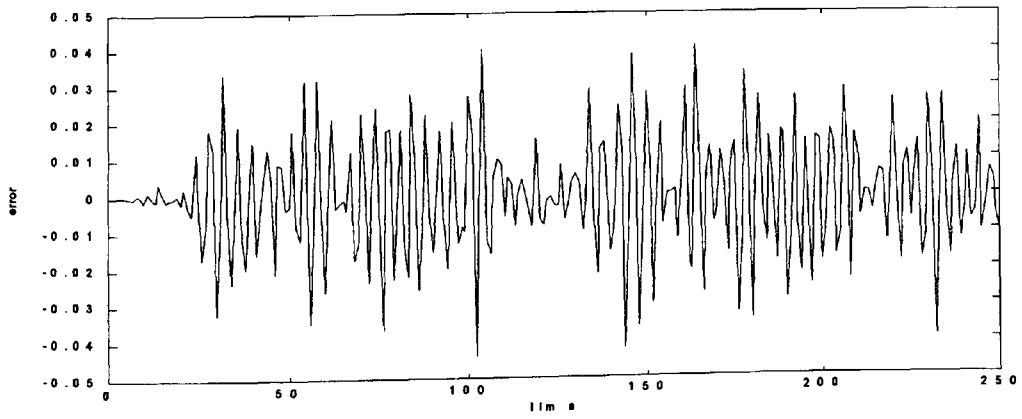

Figure 6.7 Simulink test error signal for a random input with uniform distribution [-1,1] for the 3-band NUF bank - design method 1 using N=27.

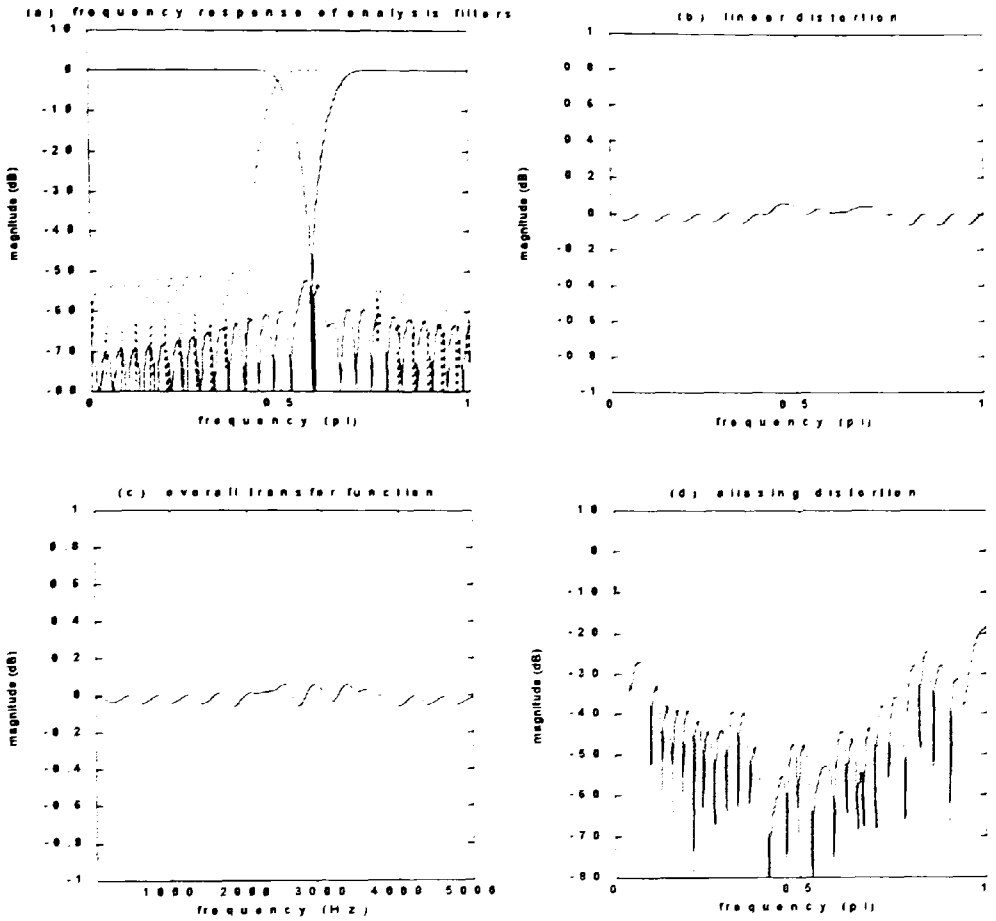

Figure 6.8 For a 3-band NUF bank using design method 1 and N=51 (a) magnitude frequency response of all analysis filters (b) transfer function of the network assuming aliasing is zero (c) overall transfer function of the network of Figure 6.1 (d) aliasing distortion.

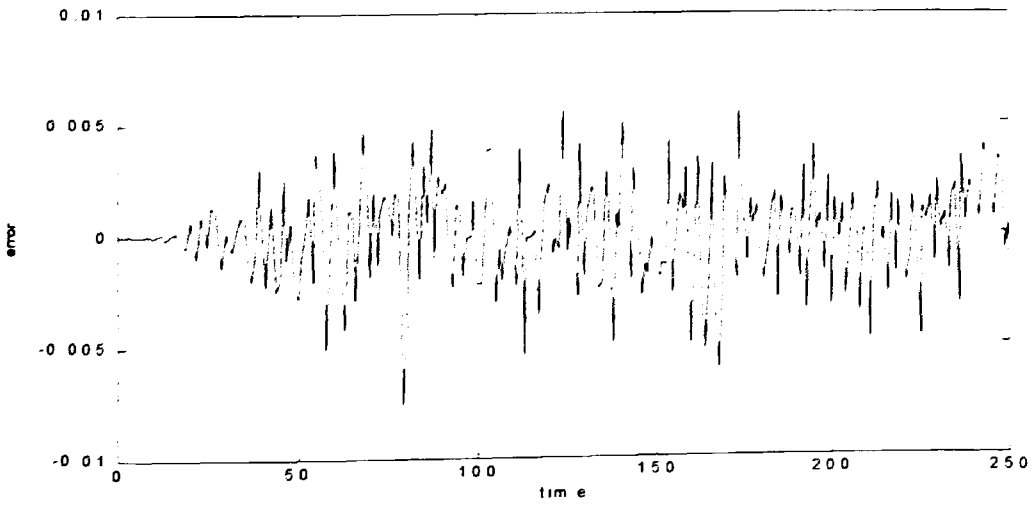

Figure 6.9 Simulink test error signal for a random input with uniform distribution [-1,1] for the 3-band NUF bank - design method 1 using N=51.

Design method 2

The results for the design method 2 for N=27 and 51 are shown in Figures 6.10 and 6.12 respectively. The corresponding Simulink test error signal results are shown in Figures 6.11 and 6.11 respectively. The tabulated results are shown in Table 6.2 where M_{pk} is the bandwidth parameter and rf_k is the roll-off factor.

Table 6.2 Results for the 3-band NUF bank using design method 2.

N	E_{pp}	E_A	M_{p0}	M_{p1}	M_{p2}	rf_0	rf_1	rf_2	v_{rms}	out_{pp}
27	0.0438	0.2458	3.9870	12.0197	5.9764	0.3229	0.9825	0.4610	0.0088	0.0606
51	0.0116	0.2568	3.9997	12.0003	6.0024	0.3255	0.9701	0.4839	0.0025	0.0193

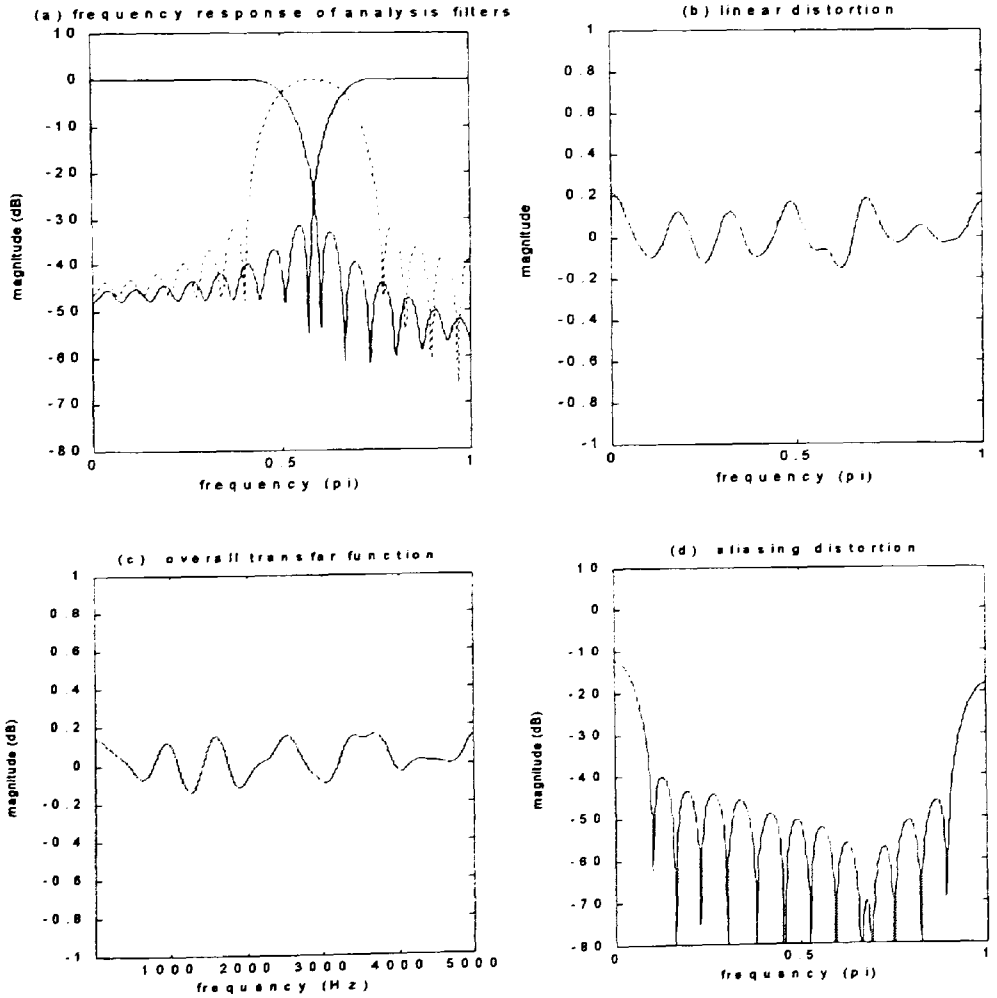

Figure 6.10 For a 3-band NUF bank using design method 2 and N=27 (a) magnitude frequency response of all analysis filters (b) transfer function of the network assuming aliasing is zero (c) overall transfer function of the network of Figure 6.1 (d) aliasing distortion.

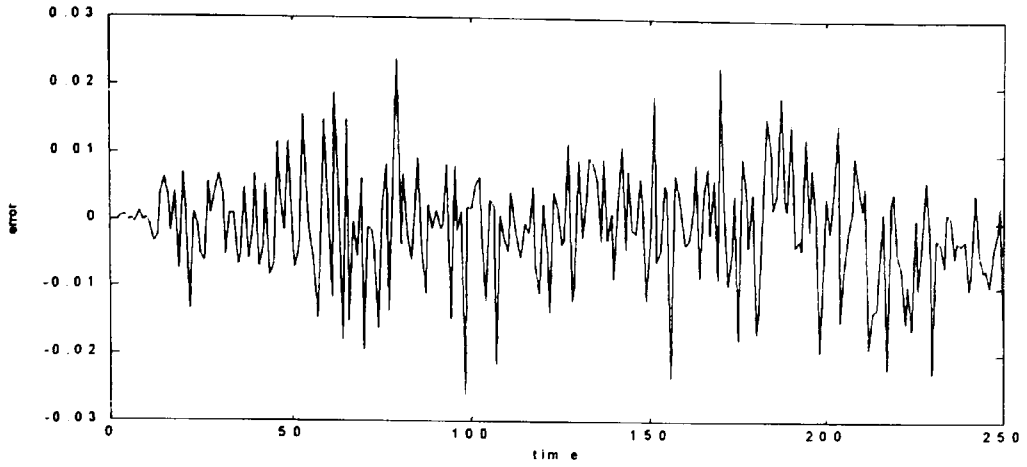

Figure 6.11 Simulink test error signal for a random input with uniform distribution [-1,1] for the 3-band NUF bank - design method 2 using N=27.

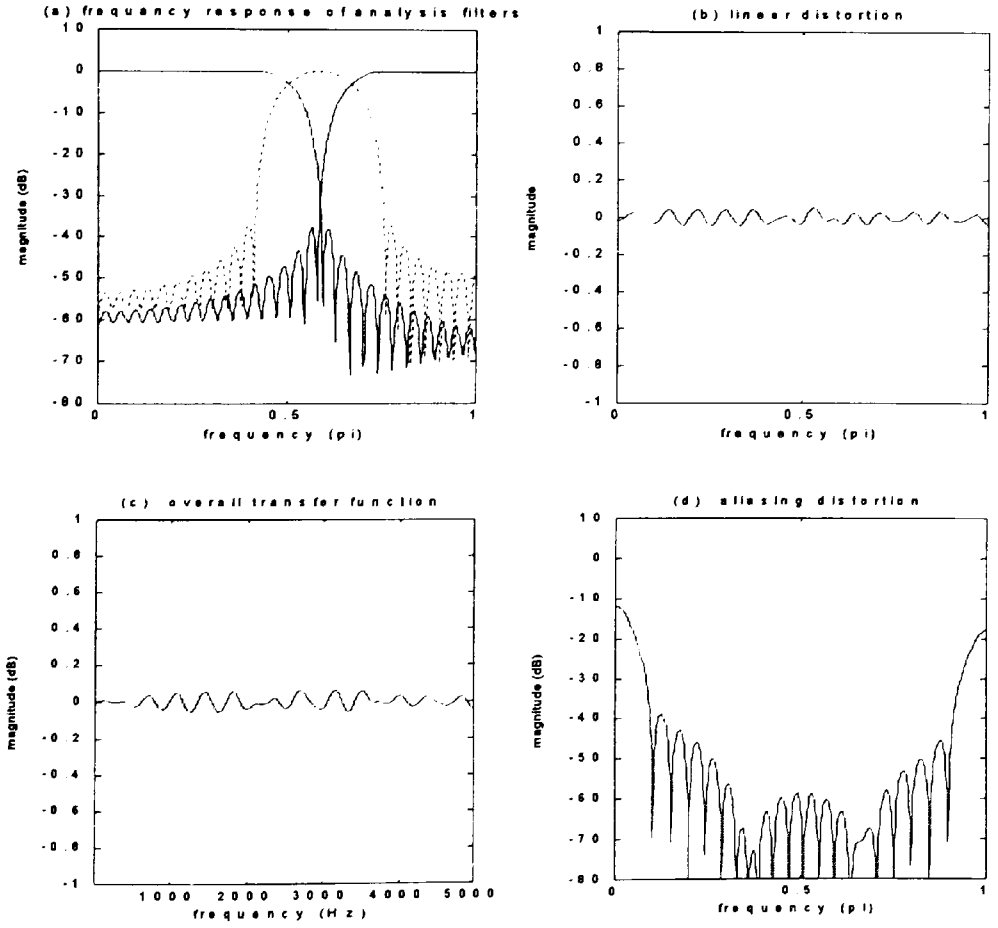

Figure 6.12 For a 3-band NUF bank using design method 2 and N=51 (a) magnitude frequency response of all analysis filters (b) transfer function of the network assuming aliasing is zero (c) overall transfer function of the network of Figure 6.1 (d) aliasing distortion.

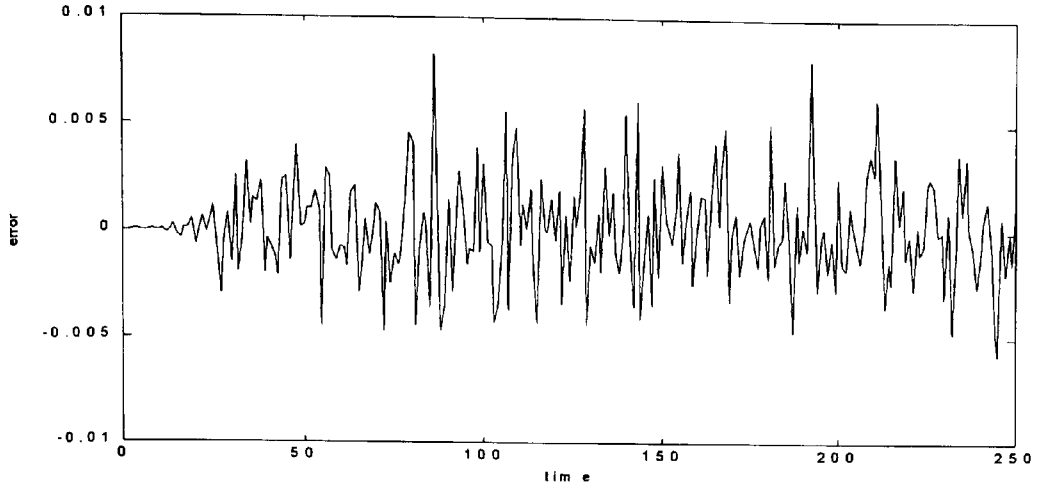

Figure 6.13 Simulink test error signal for a random input with uniform distribution [-1,1] for the 3-band NUF bank - design method 2 using N=51.

6.3.2 Design example 2: For a 5-band NUF bank – case 1

Design method 1

The 5-band non-uniform filter bank structure for case 1 is designed using r_k values of r_0=4, r_1=12, r_2=6, r_3=6, r_4=3 and M=5. The results for the design method 1 for N=45 and 65 are shown in Figures 6.14 and 6.16 respectively. The corresponding Simulink test error signal results are shown in Figures 6.15 and 6.17 respectively. The tabulated frequency perturbations of the prototype filters are shown in Table 6.3.

Table 6.3 Results for the 5-band NUF bank – case 1 using design method 1.

N	δf_{c0}	δf_{c1}	δf_{c2}	δf_{c3}	δf_{c4}
45	0.0189	0.0129	0.0183	0.0181	0.0187
65	0.0123	0.0123	0.0124	0.0122	0.0138

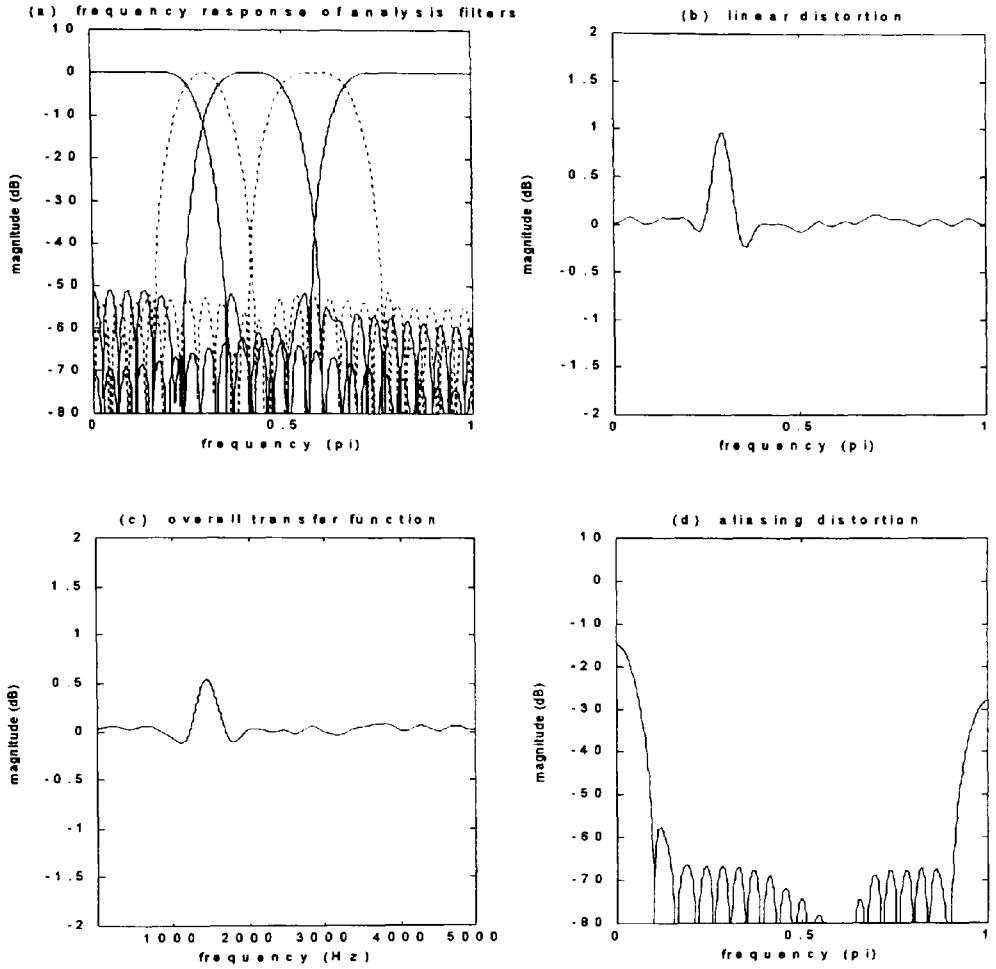

Figure 6.14 For a 5-band NUF bank – case 1 using design method 1 and N=45 (a) magnitude frequency response of all analysis filters (b) transfer function of the network assuming aliasing is zero (c) overall transfer function of the network of Figure 6.1 (d) aliasing distortion.

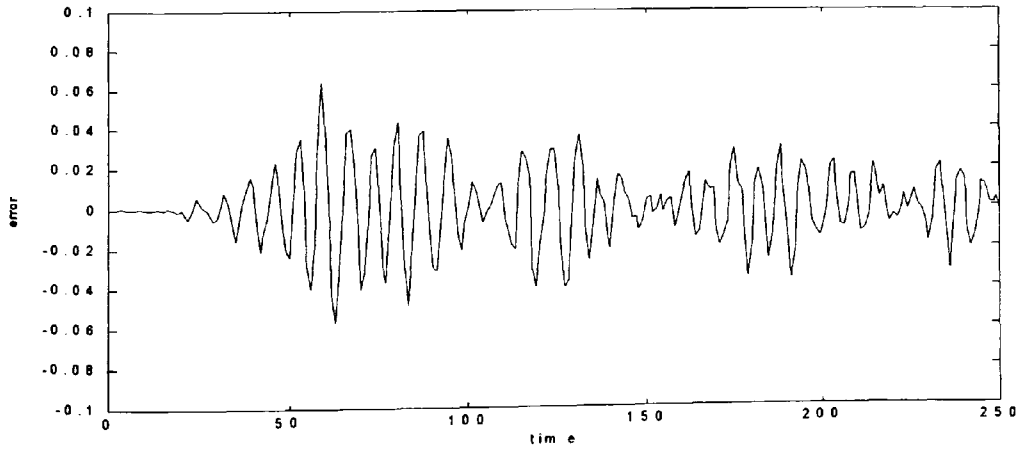

Figure 6.15 Simulink test error signal for a random input with uniform distribution [-1,1] for the 5-band NUF bank – case 1 using design method 1 and N=45.

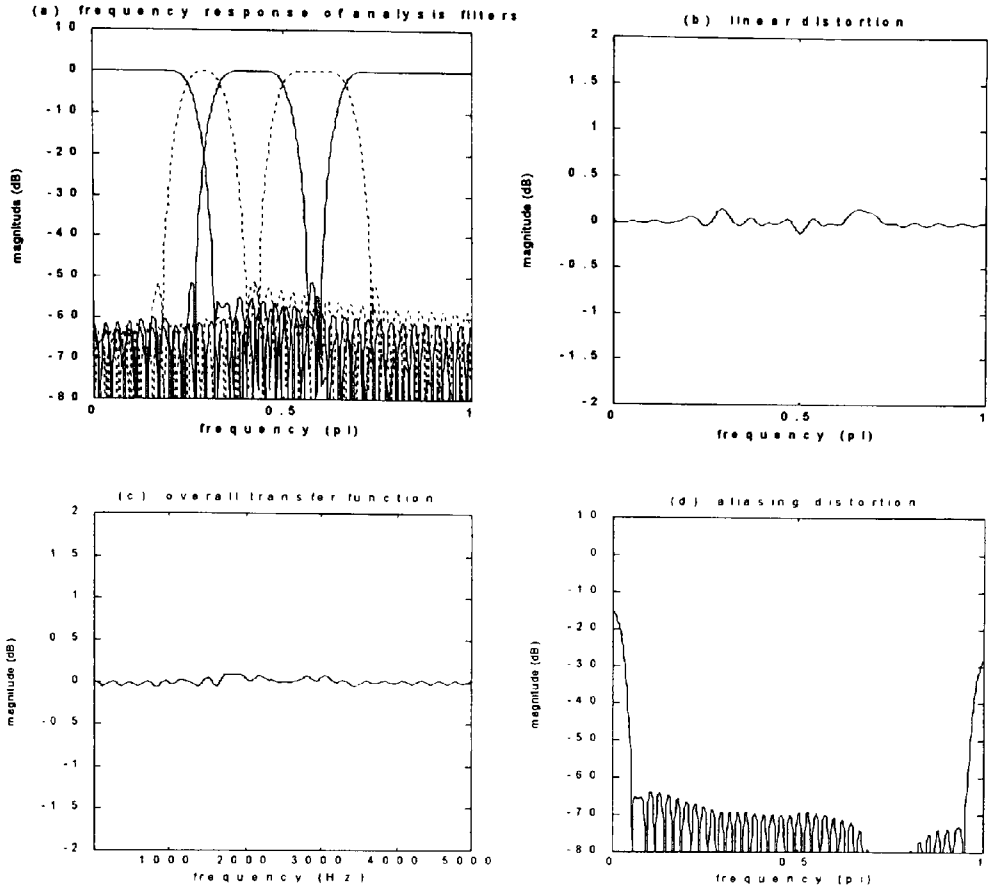

Figure 6.16 For a 5-band NUF bank – case 1 using design method 1 and N=65 (a) magnitude frequency response of all analysis filters (b) transfer function of the network assuming aliasing is zero (c) overall transfer function of the network of Figure 6.1 (d) aliasing distortion.

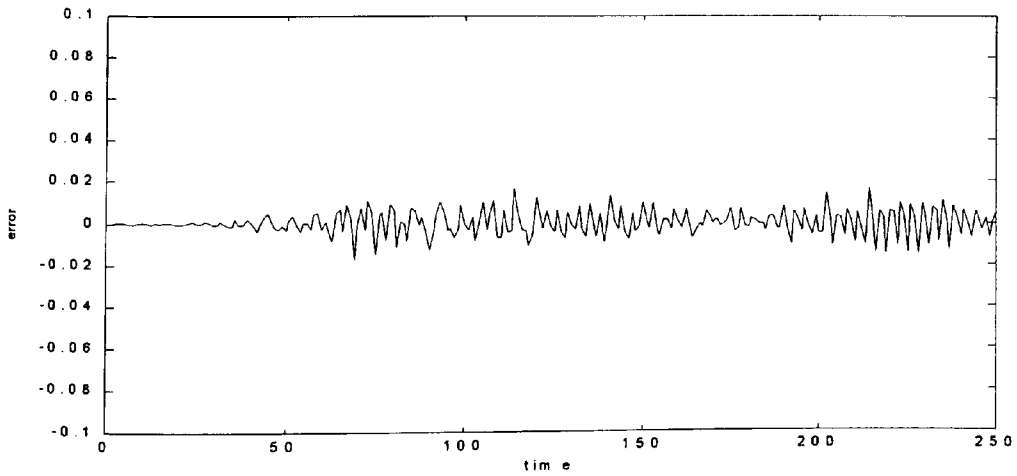

Figure 6.17 Simulink test error signal for a random input with uniform distribution [-1,1] for the 5-band NUF bank – case 1 using design method 1 and N=65.

Design method 2

The results for the design method 2 for N=45 and 65 are shown in Figures 6.18 and 6.20 respectively. The corresponding Simulink test error signal results are shown in Figures 6.19 and 6.21 respectively. The tabulated frequency perturbation parameters of the prototype filters are shown in Table 6.4 and the comparative results for design methods 1 and 2 are shown in Table 6.5.

Table 6.4 Results for the 5-band NUF bank – case 1 using design method 2.

N	M_{p0}	M_{p1}	M_{p2}	M_{p3}	M_{p4}	r_0	r_1	r_2	r_3	r_4
45	7.9539	24.3619	11.9338	11.8205	5.9719	0.3559	1.0867	0.5258	0.4807	0.2652
65	7.9985	24.0960	12.1078	11.9236	6.0084	0.2655	0.7929	0.3907	0.3921	0.2051

Table 6.5 Comparative results for the 5-band NUF bank – case 1.

N	type	E_{pp}	E_A	v_{rms}	out_{pp}
45	Design - 1	0.1441	0.1800	0.0170	0.1249
45	Design - 2	0.1051	0.1826	0.0172	0.1231
65	Design - 1	0.0312	0.1812	0.0061	0.0403
65	Design - 2	0.0489	0.1714	0.0094	0.0578

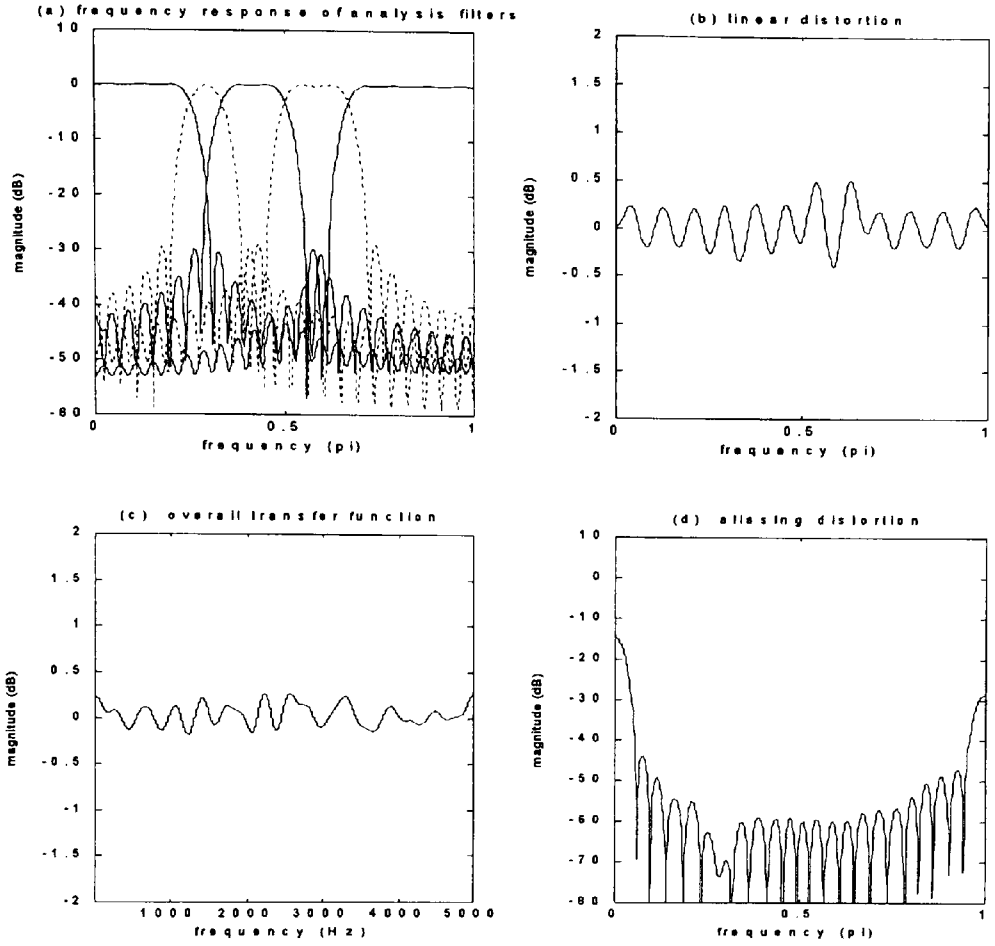

Figure 6.18 For a 5-band NUF bank – case 1 using design method 2 and N=45 (a) magnitude frequency response of all analysis filters (b) transfer function of the network assuming aliasing is zero (c) overall transfer function of the network of Figure 6.1 (d) aliasing distortion.

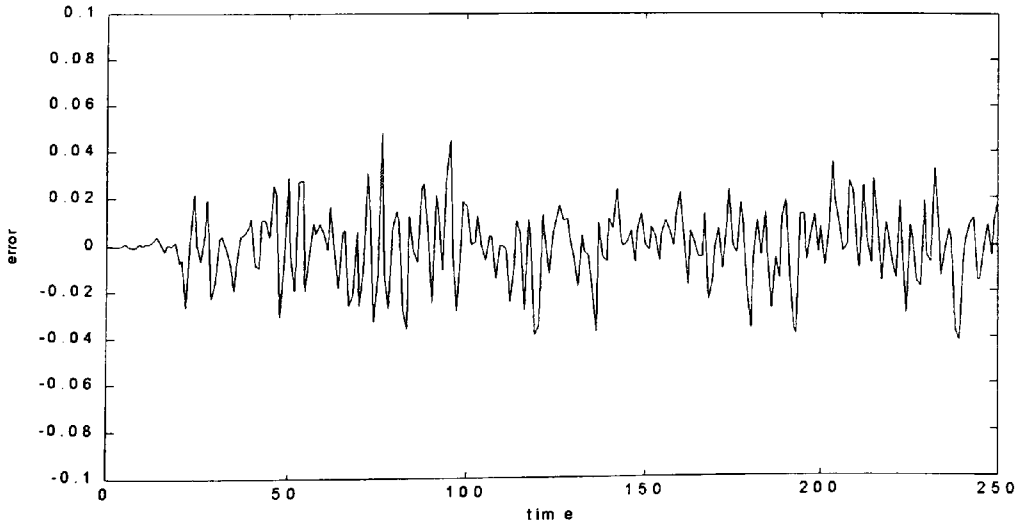

Figure 6.19 Simulink test error signal for a random input with uniform distribution [-1,1] for the 5-band NUF bank – case 1 using design method 2 and N=45.

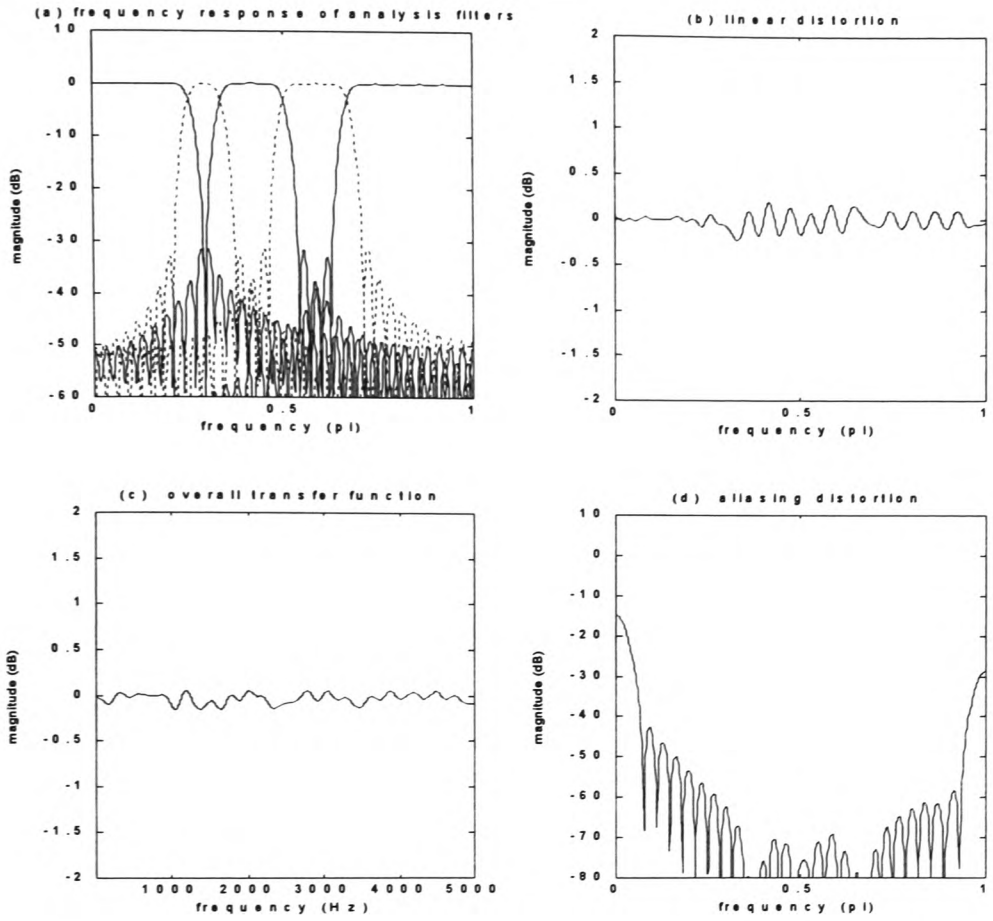

Figure 6.20 For a 5-band NUF bank – case 1 using design method 2 and N=65 (a) magnitude frequency response of all analysis filters (b) transfer function of the network assuming aliasing is zero (c) overall transfer function of the network of Figure 6.1 (d) aliasing distortion.

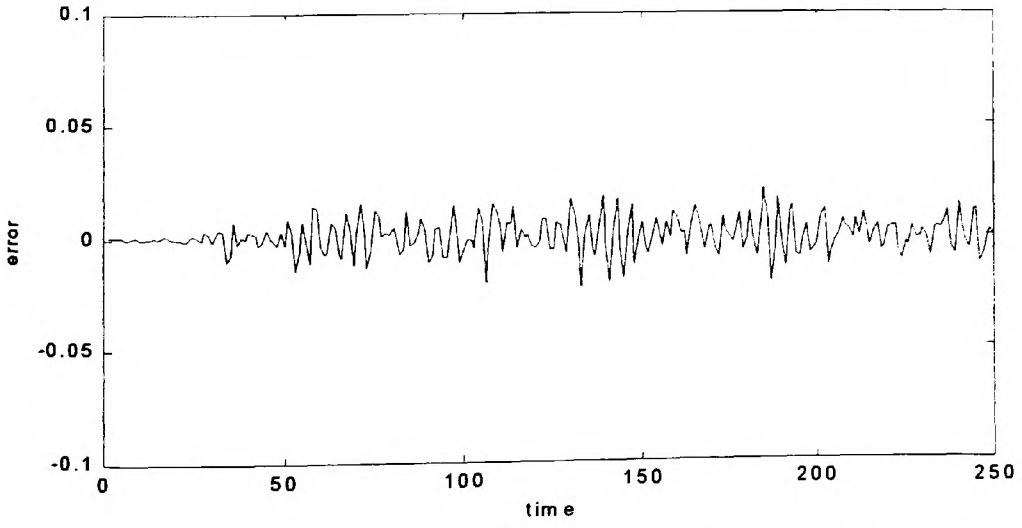

Figure 6.21 Simulink test error signal for a random input with uniform distribution [-1,1] for the 5-band NUF bank – case 1 using design method 2 and N=65.

6.3.3 Design example 3: For a 5-band NUF bank – case 2

The 5-band non-uniform filter bank structure for case 2 is designed using r_k values of: r_0=8, r_1=8, r_2=4, r_3=4, r_4=4 and M=5. The choice for this filter bank structure was based on the use of filter bands in the mixed excitation linear prediction (MELP) decoder as specified by the U.S. standard [MELP standard draft, 1998]. As part of the decoding process, a mixed excitation signal is generated as a sum of the filtered pulse and noise excitations. The band pass filter coefficients for the filtering stage are given in Appendix A of the MELP standard and is reproduced here as shown in Appendix F2.1. The study conducted here on the filter bank used for the MELP decoder is based on an analysis of the efficacy of the NUF bank simply as a system for being able to reconstruct the signal that is applied at the input. It must be recognised that identical filter bands are used for both the pulse as well as the noise excitation signals that are subsequently added to form the mixed excitation signal.

The coefficients of Appendix F2.1 are used both for the analysis filter bank as well as the synthesis filter bank for test purposes conducted in this study. The results of the amplitude and aliasing distortions are shown in Figure 6.22. The Simulink test error signal is shown in Figure 6.23. Based on design method 1, the procedure developed in this work was applied to the structure of the filter bank defined in this example. Same number of coefficients i.e. N=31 were used. The optimised results of the amplitude and aliasing distortions are shown in Figure 6.24. The Simulink test error signal is shown in Figure 6.25 and the coefficients for each analysis filter band are shown in Appendix F2.2. It must be emphasised that although the new optimised results show clear improvement when compared to the results of the MELP filter band, the tests conducted here are based entirely on the requirement for reconstructing the applied input signal. Extensive tests that are normally applied to the MELP decoder system are beyond the scope of the present work. The filter bank structure in this section was also studied using filter lengths of

N=45 and 65 and the optimised results are shown in Figures 6.26 and 6.28 for the amplitude and aliasing distortions respectively and the Simulink error results are shown in Figures 6.27 and 6.29 respectively. The tabulated results are shown in Tables 6.6 and 6.7.

Table 6.6 Results for the 5-band NUF bank – case 2 using design method 1.

N	δf_{c0}	δf_{c1}	δf_{c2}	δf_{c3}	δf_{c4}
31	0.0243	0.0280	0.0191	0.0261	0.0269
45	0.0183	0.0179	0.0181	0.0182	0.0198
65	0.0124	0.0125	0.0124	0.0129	0.0124

Table 6.7 Comparative results for the 5-band NUF bank – case 2.

N	type	E_{pp}	E_A	v_{rms}	out_{pp}
31	MELP – standard	0.6109	0.2565	0.1854	1.2930
31	Optimised – design method 1	0.1678	0.0601	0.0390	0.2497
45	Optimised – design method 1	0.0337	0.2717	0.0062	0.0446
65	Optimised – design method 1	0.0180	0.2558	0.0038	0.0251

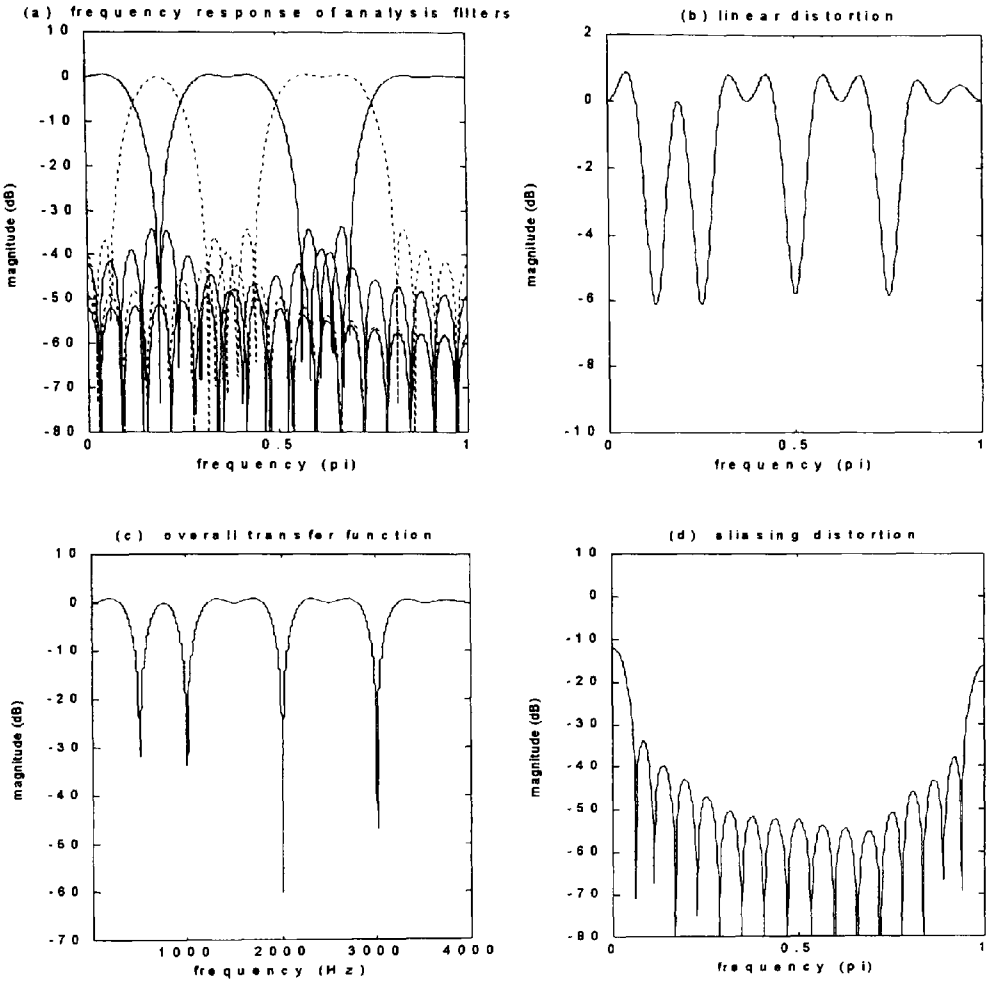

Figure 6.22 For the MELP 5-band NUF bank using N=31 coefficients given in Appendix F2.1. (a) magnitude frequency response of all analysis filters (b) transfer function of the network assuming aliasing is zero (c) overall transfer function of the network of Figure 6.1 (d) aliasing distortion.

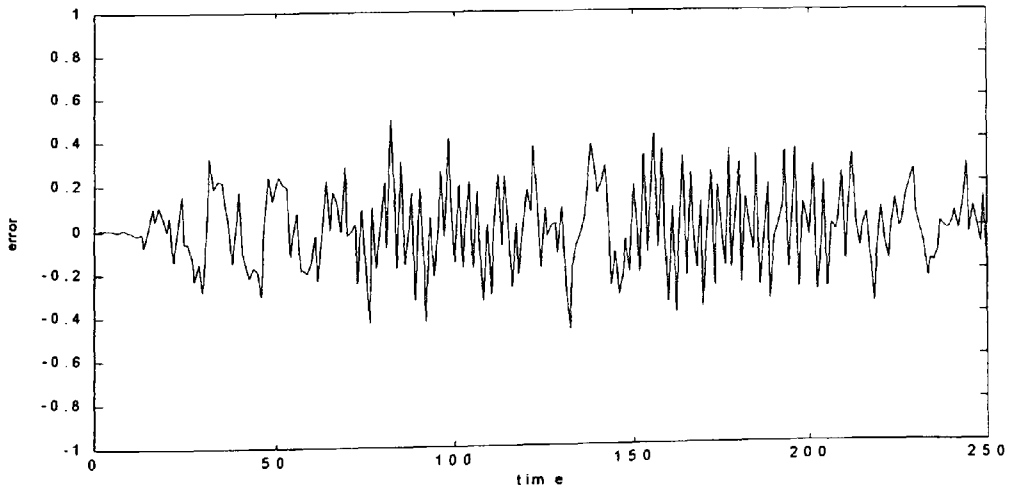

Figure 6.23 Simulink test error signal for a random input with uniform distribution [-1,1] for the MELP 5-band NUF bank using N=31 coefficients given in Appendix F2.1.

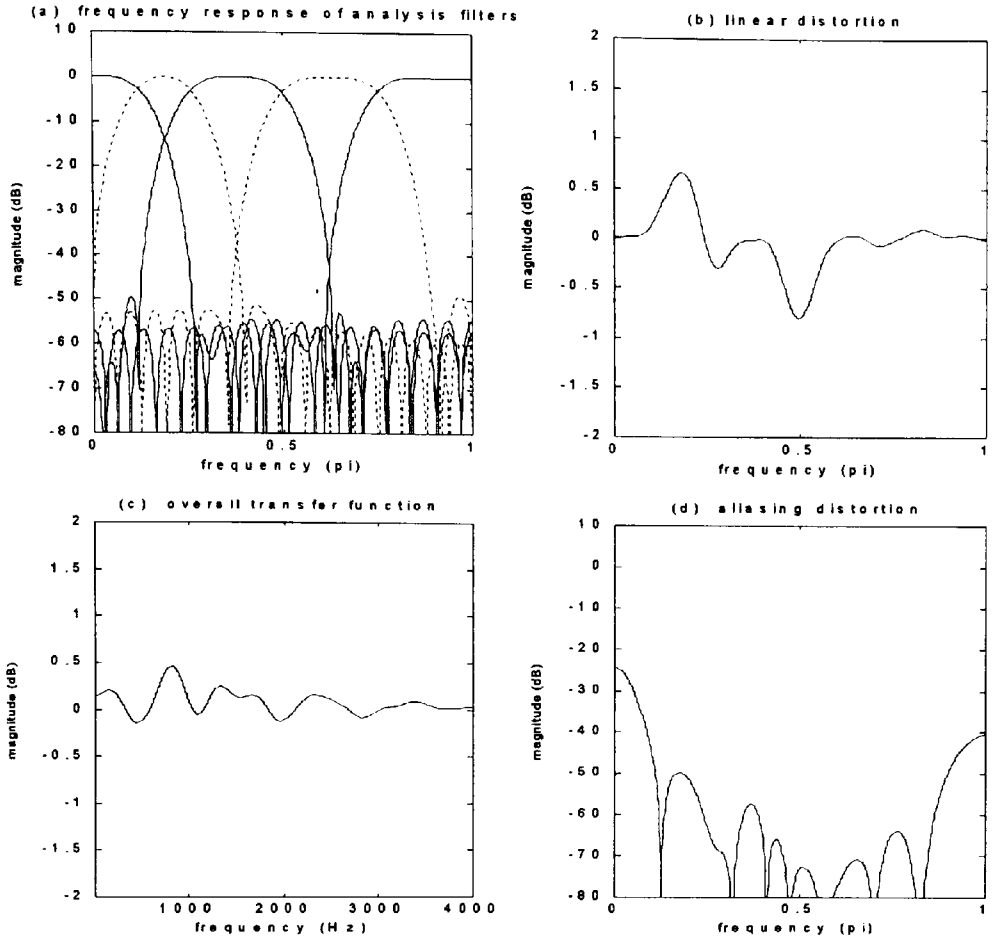

Figure 6.24 Optimised results using design method 1 for the MELP 5-band NUF bank using N=31 coefficients given in Appendix F2.2. (a) magnitude frequency response of all analysis filters (b) transfer function of the network assuming aliasing is zero (c) overall transfer function of the network of Figure 6.1 (d) aliasing distortion.

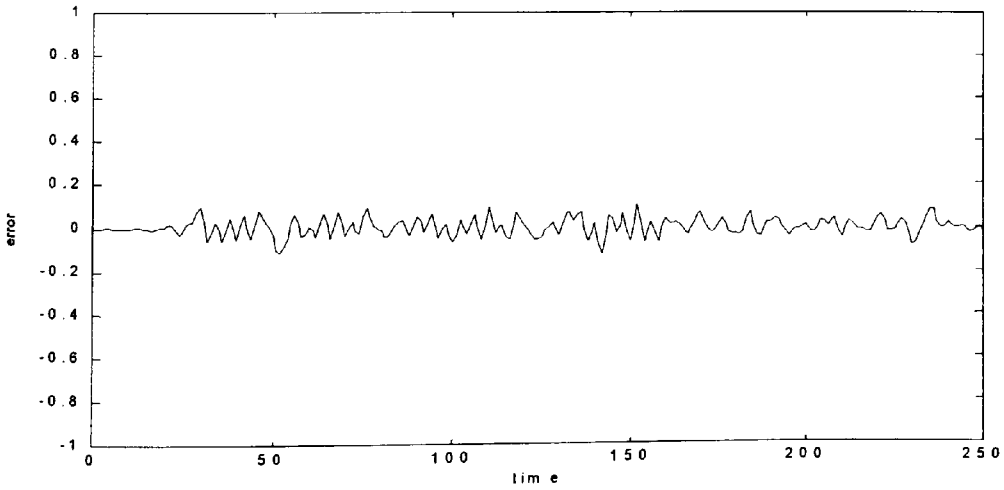

Figure 6.25 Simulink test error signal for a random input with uniform distribution [-1,1] for the optimised results using design method 1 for MELP 5-band NUF bank using N=31 coefficients given in Appendix F2.2.

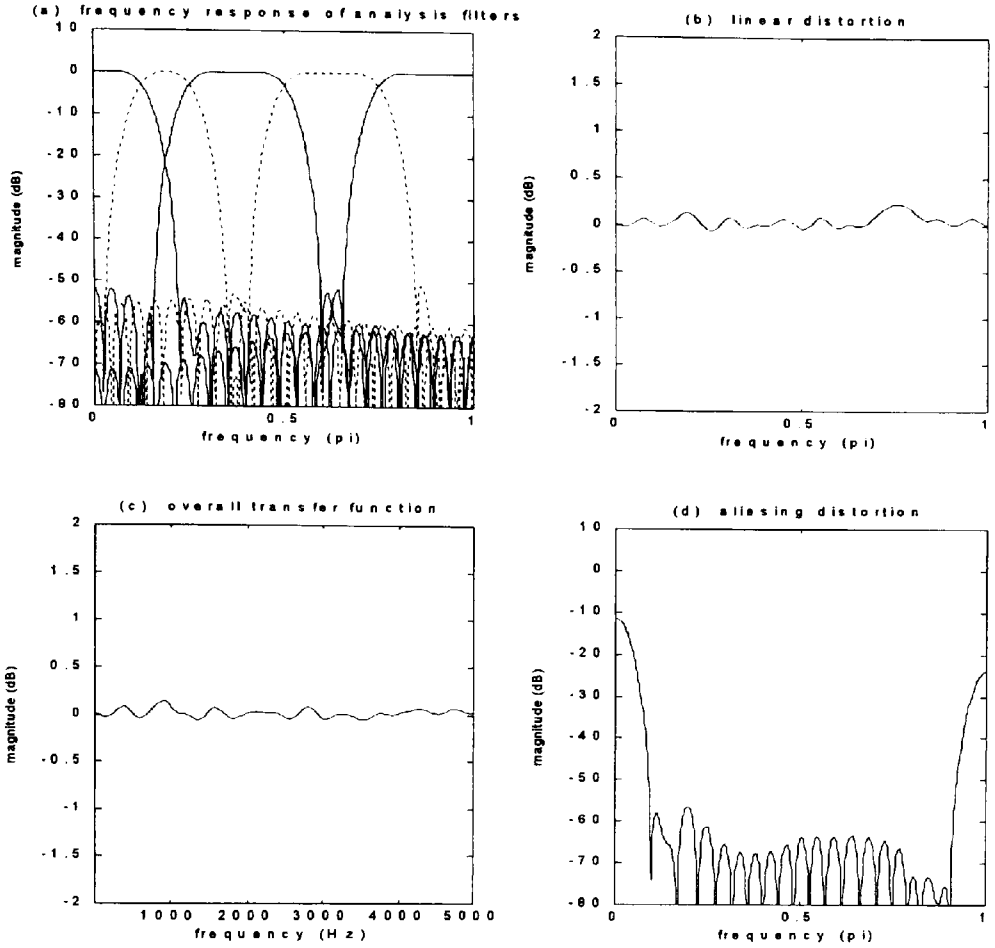

Figure 6.26 Optimised results using design method 1 for the 5-band NUF bank – case 2 for N=45. (a) magnitude frequency response of all analysis filters (b) transfer function of the network assuming aliasing is zero (c) overall transfer function of the network of Figure 6.1 (d) aliasing distortion.

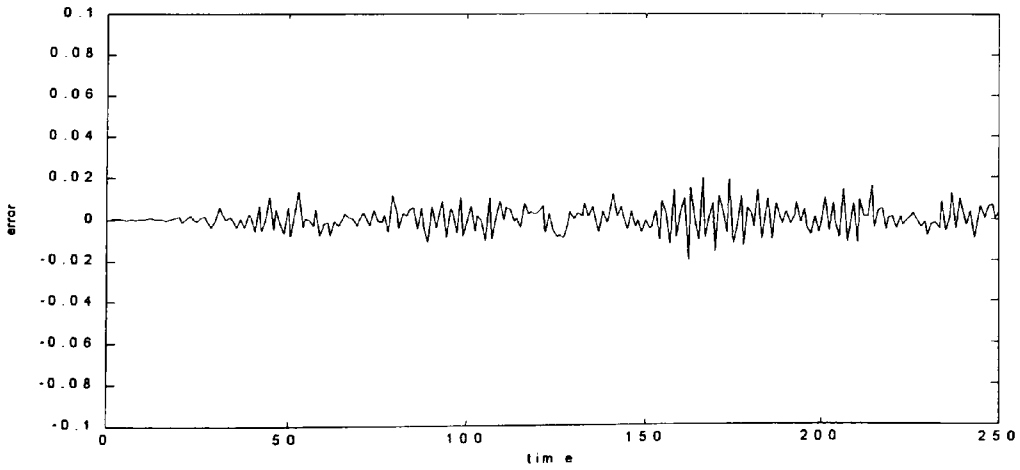

Figure 6.27 Simulink test error signal for a random input with uniform distribution [-1,1] for the optimised results using design method 1 for a 5-band NUF bank – case 2 and N=45.

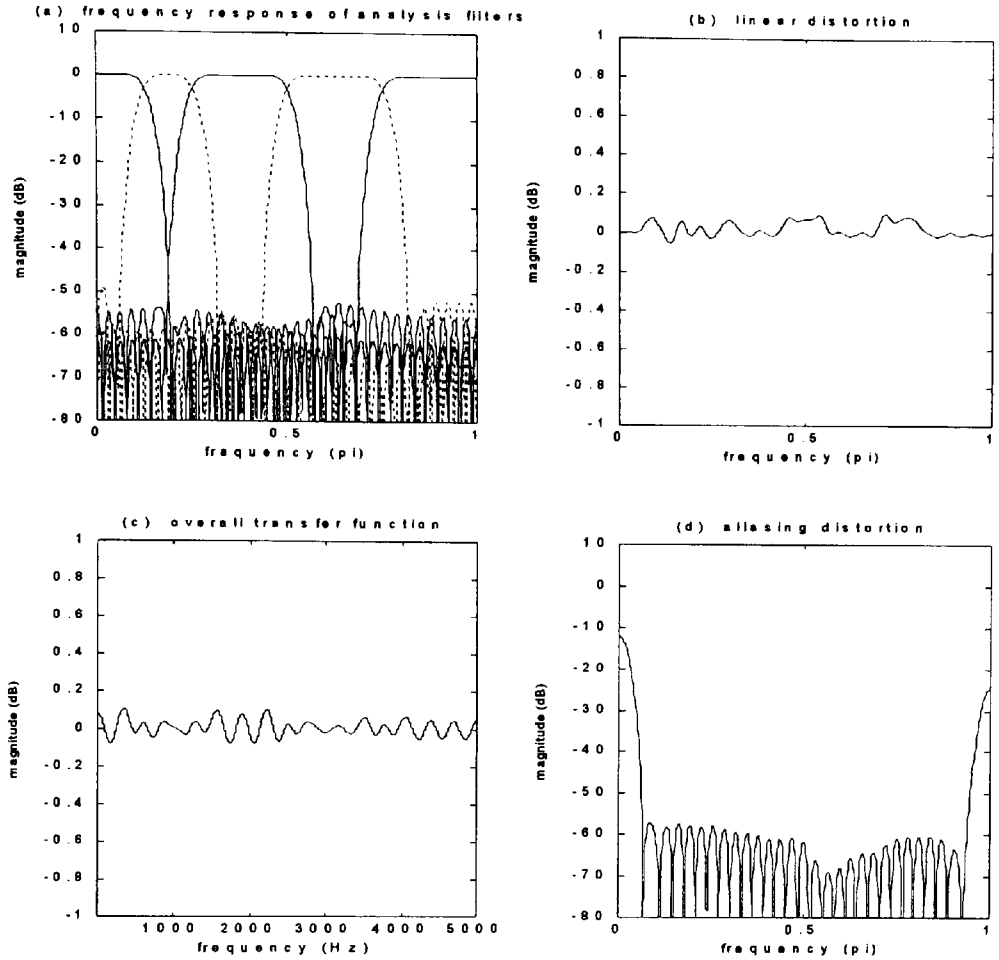

Figure 6.28 Optimised results using design method 1 for the 5-band NUF bank – case 2 for N=65. (a) magnitude frequency response of all analysis filters (b) transfer function of the network assuming aliasing is zero (c) overall transfer function of the network of Figure 6.1 (d) aliasing distortion.

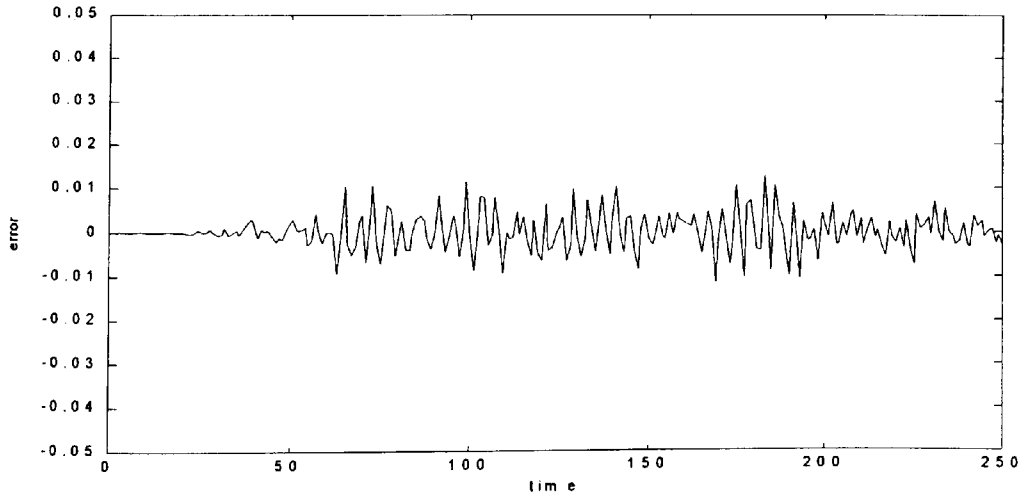

Figure 6.29 Simulink test error signal for a random input with uniform distribution [-1,1] for the optimised results using design method 1 for a 5-band NUF bank – case 2 and N=65.

6.4 Discussion of results

Two design methods gave been considered in this chapter for the implementation of the maximally decimated non-uniform filter banks where all the decimation factors are integer-valued. A number of design examples are investigated using the hybrid optimisation technique that combines the genetic algorithm search followed by the traditional downhill simplex method. The results covered in section 6.3 show significantly improved outcome through the optimisation process. The design method-1 is based on the use of multiple prototype low pass filters and their transformation by sine or cosine multiplication as developed by Chu [1985] and Wada [1995]. However, the optimisation technique developed in this work is new. The variable used to minimise the objective function is a small independent perturbation of the cut-off frequency of the individual prototype filters. This process assures the flat pass band response of the standard FIR filter as given by the `fir1.m` function of Matlab and it also retains the stop-band attenuation. A small perturbation of the cut-off frequencies will alter the bandwidths of the filter bank slightly from its specified values, however this change is fairly insignificant and should have no impact on real systems. The minimisation of the magnitude and aliasing distortions is clearly significant as is evident in the design examples covered in section 6.3. This is evident both in the magnitude and aliasing distortion graphs and the Simulink test error signal results.

The second design method considered for study in this chapter is based on the use of multiple square root raised cosine prototype low pass filters from which the analysis and synthesis filters are derived using the cosine modulation technique. The optimisation process for this method is based on the independent perturbation of the roll-off factor and the bandwidth parameters of the individual filters. This form of design and optimisation technique for non-uniform filter banks is new and no evidence of optimisation based on this method is available in literature.

A comparative study of the hybrid optimised results for the design example – 1 (i.e. for a 3-band NUF bank) and design example – 2 (i.e. for a 5-band NUF bank case 1), clearly show some disparity between design methods 1 and 2. No clear preference between the two methods has emerged. This is evident as seen in the tabulated results shown in Tables 6.1 and 6.2 for design example 1 and in Table 6.5 for design example 2. The optimised results of design example 2 for N=45 can be compared with the graphical results of Wada [1995]. Significant improvement in the linear distortion is evident for the new optimised results. However, no further comparison is possible since Wada [1995] gives no coefficient values of the prototype filters.

The design example 3 is specifically formulated to satisfy the NUF bank structure used in the MELP decoder [MELP standard draft, 1998]. The filter band coefficients are specified in Appendix A of MELP U.S. Standard and used here to derive the magnitude and aliasing distortions as shown in Figure 6.22. The Simulink test error results are shown in Figure 6.23. The optimised graphical results using design method – 1 and same number of coefficients (i.e. N=31) are shown in Figures 6.24 and 6.25. The comparative tabulated results are shown in Table 6.7. These results clearly show a significant improvement of the new optimised coefficients listed in Appendix F2.2 over the original MELP filter band.

6.5 Summary of Chapter 6 and further comments

A novel approach to the optimisation technique is proposed and investigated in this chapter where an impulse (in the form of a dirac-delta function) is applied to the entire network of Figure 6.1 and the Fourier Transform of the response is derived, giving the overall transfer function of the network. The optimisation process then minimises the difference between the maximum and the minimum value of the overall transfer function. This method leads to the

minimisation of the magnitude and the aliasing distortion in a combined manner thus giving optimal results without the need for working out optimality as a compromise between the maximum aliasing distortion and the maximum linear distortion in the absence of aliasing.

A number of severe theoretical constraints for the design of optimal maximally decimated NUF banks are relaxed in favour of a direct design approach based on the use of hybrid optimised low pass FIR prototype filters. Two direct design methods have been studied. Method one is based on the use of standard FIR low pass prototypes and their transformation using sine or cosine multiplication. The optimisation technique used for this method is based on marginally perturbing the cut-off frequencies of the prototype filters independently to minimise the objective function. The second design method is based on using a square root raised cosine FIR low pass prototype filters. The analysis and synthesis filters are derived from the prototype filters by the cosine modulation technique. The optimisation process for this design method is based on independently perturbing the roll-off factor and the bandwidth parameters of the prototype filters. Although the non-uniform filter banks considered in this study are restricted to the type using integer-valued decimators, this study can be easily extended to the case of NUF banks using rational valued decimators.

The main contributions of this part of the study are the following.

- Real-valued genetic algorithm codes have been developed for the optimisation of the non-uniform M-channel maximally decimated filter banks using integer decimators. Two design methods were considered. The first method uses transformation of low-pass prototype filters by sine or cosine multiplication and the second method is based on the cosine modulation technique. Further modification of the codes involved inclusion of the Simplex code for comparative and hybrid optimisation study.

- The hybrid optimisation process is applied to the entire network of the non-uniform filter bank. The variables are the cut-off frequency of the prototype filters for design method one and the bandwidth and the roll-off factors for design method two. The variables are independently perturbed for the respective design methods and the overall transfer function of the filter bank network is optimised more minimal errors. The minimisation of the magnitude and aliasing errors is thus achieved in a combined manner. Significantly improved new results using the GA and hybrid optimisation are given in Tables 6.1, 6.2, 6.5 and 6.7.

Several design examples have been considered to demonstrate the potency of the design and optimisation techniques developed in this study. The specific and some comparative results further substantiate this. The use of a hybrid optimisation process has demonstrated a strong synergy between the regulated random chance based technique of genetic algorithms that work well over a wide landscape of possible discontinuous functions and a standard minimisation algorithm such as down hill Simplex, that works well for largely continuous functions.

Chapter 7: Conclusions and further work

Overview of Chapter 7: This chapter draws a summary, the contributions and concluding remarks of the main focus of study that is discussed, investigated and reported in this thesis. Some suggestions for further work are also included.

7.1 A summary and the contributions

The main focus and contribution of this thesis is the study and application of the genetic algorithm optimisation method in the area of digital filters and multirate filter banks. For the case of digital filters, the optimisation was conducted for finite word-length coefficient constraints that would inevitably lead to changes of the transfer function of the filters. Both, finite and infinite-impulse response digital filters have been considered for optimisation. For the case of multirate filter banks the design issues and their optimisation of three types of structures is considered. These are; a 2-channel quadrature mirror filter, a multiple M-channel uniform filter bank and a multiple M-channel non-uniform filter bank. Where possible, the GA optimised results are compared with the results obtained using alternative optimisation methods.

Chapter 2 covers GA optimisation for ten FIR band select filters. A comparison is drawn with the results taken from [Kodek and Steiglitz, 1981] that were optimised using the integer programming method. The following contributions are claimed with respect to the outcome of results covered in Chapter 2.

Contribution 1: A real integer-valued genetic algorithm code has been developed for the optimisation of finite word length constrained coefficients of FIR digital filters. The new GA optimised results are significantly superior when compared with the integer programming

method optimised results taken from [Kodek and Steiglitz, 1981]. The comparative results are shown in Tables2.4 and 2.5 and the new coefficient values are listed in Table 2.3. Further comparison of the GA optimised results was conducted with the simple hill climber algorithms. The results for a selection of the FIR digital filters are shown in Table 2.7.

Contribution 2: The GA optimised results for FIR filters demonstrate the assertion that the maximum deviation derived using statistical methods [Chan and Rabiner, 1973] as given by Equations 2.25 and 2.26 holds well as is evident from the GA optimised results seen in Figures 2.6 and 2.8.

In Chapter 3, the finite word-length coefficient optimisation for IIR filters both of the direct form and of the second order cascade form structures was covered. Due to the recursive nature of such filters, there is a distinct possibility that the FWL coefficient constraint could cause an otherwise stable filter to become unstable. The condition for assuring stability of the final optimised design imposes an extra constraint on the optimisation process. There is also an issue about the susceptibility to changes in the transfer function for different IIR filter structures. Some work in this topic has been previously reported in [Harris and Ifeachor 1995, 1998] and [Arslan and Horrocks, 1995]. However, no quantifiable metric for a comparative study was available. For this reason, a set of IIR filters are proposed and listed in Table 3.1. GA optimisation tests were conducted for these filters for different order values and number of bits representing the FWL coefficients. The outcome of the tests is listed in Tables 3.2 and 3.3 for the direct form and the second order cascade form structures respectively. The following contribution for the study in Chapter 3 is thus claimed.

Contribution 3: Real integer-valued genetic algorithm codes have been developed for the optimisation of the finite word length constrained coefficients of IIR digital filters. The direct

form and the second order cascade form structures have been considered. The GA optimised results for different filter orders and number of bits representing the coefficient values are seen to be vastly superior when compared with the simply rounded coefficient value results. These results are shown in Tables 3.2 and 3.3 and the actual coefficient values are listed in Appendices C1.2 and C2.2. Further comparison of the GA optimised results was conducted with the simple hill climber algorithms. The results for a selection of the IIR digital filters are shown in Table 3.4.

Multirate processing of digital signals is an important area that has significant applications in digital audio systems and in speech and image processing. A basic form of a multirate system is the quadrature mirror filter bank with specific applications in sub-band coding and data compression of speech signals. The issues of design, optimisation, simulation and real-time implementation of a specific class of a QMF bank [Tay, 1998] are considered in Chapter 4. The GA optimisation is applied in two stages for the final realisation of the design. In the first stage, the issues of design optimisation are considered. While in the second stage, issues relating to finite word-length coefficient optimisation of filters using genetic algorithms as developed in the earlier chapters are applied for real-time implementation on a target DSP hardware system.

In order to test the potency and robustness of the GA method, a new 'creep' code was developed that performed a 'tumbling like' minimisation algorithm. This code was applied to the first stage design optimisation and the results were compared with results obtained using the standard gradient and non-gradient based minimisation methods. These comparative results are shown in Tables 4.1, 4.2 and 4.3 respectively for the three design examples studied. For the design stage the new GA 'creep' code performs well in the optimisation cycle when compared with the direct form GA code, as is evident from Figures 4.8 and 4.9. However, when compared with a hybrid method of GA optimisation followed by a standard quasi-Newton or downhill Simplex method,

then the hybrid scheme generates significantly improved results and is also more efficient. Furthermore, the new optimised results show significant improvement when compared with the original design results of Tay [1998] as seen in Tables 4.1, 4.2 and 4.3 for three design examples respectively.

The following contribution is thus claimed for this part of the study.

Contribution 4: A real-valued genetic algorithm code has been developed for the optimisation of the design of a class of quadrature mirror filter bank that has a perfect reconstruction property. This code was further enhanced to include a 'creep' code option within the main GA code that uses a 'tumbling-like' minimisation algorithm. The new 'creep' code was developed to draw a comparative study with the standard quasi-Newton and Simplex optimisation methods. The new GA hybrid optimised results show a significant improvement when compared with the original design results as seen in Tables 4.1, 4.2 and 4.3.

For the real-time implementation of the optimised design, several issues are relevant for discussion and are listed here.

- The design of the QMF bank develops IIR filters for the sub-bands, both analysis and synthesis filters and has a perfect reconstruction characteristic.
- The transfer function of the IIR filters is of the type that can be decomposed directly into a computationally efficient polyphase form.
- For telephone quality speech signal coding and compression, the polyphase components are optimised using 8-bit finite word-length coefficients. The GA optimised magnitude responses are shown in Figure 4.24.

- An idea of companding for the QMF bank is introduced as seen in Figure 4.26. This structure is tested for a number of sub-band quantised signal values using different number of bits. The real-time testing and comparisons were drawn using the Mean Opinion Score metric. The results for a sample of individuals are shown in Table 4.9.

The optimised design from stage one was implemented on a real-time system. For this the transfer function of the IIR filters were firstly decomposed into the computationally efficient polyphase components and then FWL coefficients were optimised using genetic algorithms. An idea of companding for the QMF bank is introduced as seen in Figure 4.26. The real-time tests were conducted on a digital signal processing starter kit based on the fixed point TMS320C50 processor. The tests using the Mean Opinion Score show a trend towards improvement for the 'with companding' option as seen in Table 4.9. The sample of individuals used is small and thus statistically not significant. The contribution for this part of the study is as follows.

Contribution 5: The new GA optimised design of the QMF bank was implemented on a real-time TMS320C50 digital signal processing starter kit. Tests were conducted using the Mean Opinion Score metric that showed an improvement of results using the 'with companding' option as proposed in Figure 4.26.

The study of a two-channel filter bank as discussed in Chapter 4 was extended to the case of a uniform M-channel filter bank in Chapter 5. This form of a multiple channel filter bank has a potential for improved signal coding efficiency and compression gain. The design and optimisation of the uniform M-channel filter bank is based on the cosine modulation technique. For this, all the sub-bands of the filter bank are derived from a single low-pass prototype filter. The overall optimisation of the filter bank is then based on the appropriate choice of the prototype filter parameters. A square root raised-cosine form of the low-pass prototype filter

that has a linear phase FIR characteristic was used for this application. For this form of filter, two parameters i.e. the bandwidth and the roll-off factor were marginally perturbed to generate improved results of the overall filter bank. The optimisation process was based on the use of genetic algorithms and the standard gradient and non-gradient based methods.

The GA and hybrid optimised results for the design of a uniform, maximally decimated, 8-channel filter bank for the design example 1 as seen in Table 5.1 shows a substantial improvement when compared with the optimised results of a non-linear optimisation package that is reported in [Vaidyanathan, 1993]. Furthermore, the new optimised results for design example 2 as seen in Table 5.4 for N=141 (i.e. number of filter coefficients) show a significant improvement over the graphical results of Fliege [1994] as given in Table 5.6(d) for N=257. It is also instructive to note that the improvement of the optimised results between design example 2 for N=141 and the design example 3 for N=257 is fairly small. This observation has clear implications for the designer in terms of accuracy of design implementation and computational overheads of throughput. The major contribution of this part of study is the following.

Contribution 6: A real-valued genetic algorithm code has been developed for the optimisation of a uniform maximally decimated M-channel filter bank. Further modification of this code involved inclusion of the quasi-Newton and the Simplex codes for a comparative and hybrid study. Significantly improved new results using the GA and hybrid optimisation are given in Tables 5.1, 5.4 and 5.6.

The study of a uniform M-channel filter bank was extended to the case of a non-uniform M-channel filter bank and is covered in Chapter 6. This form of a filter bank has a potential for matching the spectral sub-bands more closely to the ensemble average of the energy of real signals. High coding gain with minimal loss of spectral quality is thus achievable. However,

extensive theoretical and practical design constraints exist for achieving the perfect reconstruction characteristic of the non-uniform filter banks. The problem is then reduced to relaxing constraints in favour of ease of design and then using optimisation techniques for minimising the errors. Two methods of direct design based on using FIR low-pass prototype filters were investigated in this study. The first method is based on the transformation of the low-pass prototype by sine or cosine multiplication and the second method is based on the cosine-modulated pseudo QMF bank method. The second method of design is an extension of the design procedure used in Chapter 5 for uniform M-channel filter banks for which multiple prototype low pass filters are used to achieve appropriate frequency band shifts.

The hybrid optimisation process using a genetic algorithm and the Simplex method is applied to the network of a non-uniform filter bank. The minimisation of the magnitude and aliasing distortion is achieved in a combined manner by working out the overall transfer function of the network. This is achieved by marginally perturbing independently the cut-off frequencies of the prototype filters for the case of design method one. For the second design method, the bandwidth and the roll-off factor parameters of the square root raised-cosine prototype filters are independently perturbed. The optimised results for a number of design examples are shown in Tables 6.1, 6.2, 6.5 and 6.7. The main contributions of this part of the study are stated as follows.

Contribution 7: Real-valued genetic algorithm codes have been developed for the optimisation of the non-uniform M-channel maximally decimated filter banks using integer decimators. Two design methods were considered. Further modification of the codes involved inclusion of the Simplex code for comparative and hybrid optimisation study. The hybrid optimisation process is applied to the network of the non-uniform filter bank. The minimisation of the magnitude and

aliasing errors is thus achieved in a combined manner. Significantly improved new results using the GA and hybrid optimisation are given in Tables 6.1, 6.2, 6.5 and 6.7.

7.2 Suggestions for further work and conclusions

There are clearly a number of issues that have remained unanswered and others that have only partially been resolved. These are thus good candidates for further research and investigation. Some of these issues are discussed here.

- *Categorisation of objective function landscapes*

  An important area for investigation towards an effective means for GA optimisation is an understanding of the objective landscapes and their categorisation on the basis of their relationship with the GA and hybrid regimes. Most applications of GAs are conducted by using trial parameters based either on intuitive initial trials or on prior knowledge of similar problem situations and their solution. A good understanding of the GA and hybrid process and its relationship with the objective function landscape will clearly help in the robust and effective optimisation process.

  A starting point for the investigation into this problem must be a detailed description or display of the objective function landscape profile. This is not an easy problem to articulate due to the multidimensional nature of the objective function used in many engineering related applications. Some insight into this problem can, however, be gained by simplifying the objective function into a series of three-dimensional functions. An objective function profile then begins to emerge. The next stage is to investigate the objective function in relation to the GA parameters. A simple test for the population size and mutation rate against the objective function was conducted in Chapter 4, Figure 4.7 of this study. However, other GA parameter types such as fitness function, crossover and selection were

not tested. Each of these could have a bearing on the effectiveness of the GA process. Of wider interest here is an issue about broad categorisation of the objective function landscapes of target applications for the effective and efficient GA process.

Optimal telephony speech signal compression using a non-uniform filter bank

The optimised speech signal coding and compression using a uniform 2-channel filter bank was considered in Chapter 4 of this study. Further improvement of the compression gain is achievable using an optimised non-uniform filter bank. The methods of FIR filter finite word length optimisation developed in Chapter 2 and the design optimisation methods of non-uniform filter bank developed in Chapter 6 can be applied for this purpose. Furthermore, companding issues considered in Chapter 4 can be extended and applied to this application with the purposeful aim of improving the mean opinion score metric for the quality of compressed telephony speech signal.

A commonly used structure of the non-uniform filter bank for this application is based on a five-band split of the input signal given by 0 – 0.5, 0.5 – 1.0, 1.0 – 2.0, 2.0 – 3.0, 3.0 – 4.0 kHz when the sampling frequency is 8 kHz. A number of quantisation schemes are recommended [Porat, 1997 pp. 497]. This structure of the filter bank is identical to the one considered as design example 3 in Chapter 6 (section 6.3.3). It must also be noted that sub-band coding is applied in the compression of the high-fidelity audio signals. The study covered in Chapter 6 can also be extended to this application. A typical bit rate for hi-fi audio is about 700 kbits per second per channel. The MPEG standard for compression defines 32 sub-bands for which a compression ratio of 5.5 is achievable. This generates a bit rate of 128 kbits per second per channel for almost compact disc like music quality.

- *Optimisation of non-uniform filter banks using rational number decimation factor*

  This problem is a direct extension of the issues considered in Chapter 6 with the exception that instead of using integer valued decimation factors, rational number decimation is used. The use of rational numbers as decimators can give further refinement to the sub-band frequency widths of the filter bank. This means that a closer approximation to the ensemble average of the energy of real signals can be matched. The theoretical issues of design of these type of filter banks so that the errors due to magnitude, phase and aliasing can be minimised has been previously considered and reported in literature [Wada, 1996], [Argenti and Del Re, 1998]. Of interest here is to use the optimisation methods developed in Chapter 6 and to seek for improved results.

- *Use of parallel GAs in the optimisation of digital filters and multirate filter banks*

  The development of parallel GAs is an important area that has seen some applications in the field of digital signal processing [Xu and Daley, 1995], [Kwong and Chan, 1997]. The parallel GA is composed of several sequential algorithms operating simultaneously. The main aspect of linkages between the sequential algorithms is by means of the genetic operation of migration. For the case of a FWL coefficient digital filter then the chromosome is defined by a set of coefficients representing the specific filter. The distinguishing feature of migration can then be implemented by transferring a copied version of the distinct set of filter coefficients with minimum error function in each local population to the neighbouring node. This procedure generally enhances the performance of the algorithm. This process can also be extended to the case of multirate filter banks and their performance tested against those of the simple GA.

Finally in conclusion, the use of genetic algorithms as an effective and robust optimisation tool in the field of digital filters and multirate filter banks has been substantially demonstrated in this

study. For the specific area of FWL coefficient constraints of digital filters, genetic algorithms alone are shown to have generated superior results when compared with other methods of optimisation reported in literature. However, for the design optimisation of multirate filter banks, genetic algorithms performed well in a hybrid format. This was achieved by conducting an initial search over a wide landscape using genetic algorithms and then 'homing-in' on the best individuals by using recognised optimisation tools such as the quasi-Newton or the Simplex method.

References

Argenti F. and Del Re E. (2000). Design of biorthogonal M-channel cosine-modulated FIR/IIR filter banks. *IEEE Transactions on Signal Processin.* vol. **48**(3), pp. 876-881.

Argenti F., Brogelli B. and Del Re E. (1998). Design of Pseudo-QMF banks with rational sampling factors using several prototype filters. *IEEE Transactions on Signal Processing,* vol. **46**(6), pp. 1709-1715.

Argenti F. and Del Re E. (1996). Non-uniform filter banks based on a multi-prototype cosine modulation. *Proceedings of IEEE International Conference on ASSP*, Atlanta, GA. May 1996.

Arslan T. and Horrocks D.H. (1995). A genetic algorithm for the design of finite word length arbitrary response cascaded IIR digital filters. *Proceedings of IEE International conference on Genetic Algorithms in Engineering Systems: Innovations and Applications (GALESIA, 1995),* pp.276-281, 1995.

Avenhaus E. (1972). On the design of digital filters with coefficients of limited word length. *IEEE Transactions on Acoustics, Speech, Signal Processing.* vol. **20**, pp.206-212, 1972.

Baicher G.S. and Sherrington J.A. (1996). Learning about digital signal processing using spreadsheets and simulation software. *IEE Journal on Engineering Science and Education.* vol. **5** (1), pp. 41-48, 1996.

Baker J.E. (1987). Reducing bias and inefficiency in the selection algorithm. *Proceedings of the International conference on genetic algorithms (ICGA-2),* pp.14-21, Lawrence Erlbaum Associates Publishers, 1987.

Basu S., Chiang C.H. and Choi H.N. (1995). Wavelets and perfect reconstruction subband coding using causal stable IIR filters. *IEEE Transactions on Circuits and Systems II: Analog Digital Signal Processing.* vol. **42**(1), pp. 24-38.

Bellamy J. (2000). *Digital Telephony* – 3rd edition, New York: John Wiley.

Bright M.S. and Arslan T. (2001). Synthesis of low-power DSP systems using a genetic algorithm. *IEEE Transactions of Evolutionary Computation*. vol. **5**(1). February 2001.

Bull D.R. and Horrocks D.H. (1991). Primitive operator digital filters. *IEE Proceedings part.* vol. **138**(3), pp.401-412.

Censor Y. (1977). Pareto optimality in multiobjective problems. *Applied. Math Optimization.* vol. **4**, pp.41-59.

Chan D.S.K. and Rabiner L.R. (1973). Analysis of quantization errors in the direct form for finite impulse response digital filters. *IEEE Transactions on Audio and Electroacoustics*. vol. **21**(4), pp.354-366.

Chen T. and Francis B.A., (1995). Design of Multirate Filter Banks by H∞ Optimization. *IEEE Transactions on Signal Processing*. vol. **43**(12). Dec 1995.

Chipperfield A., Fleming P., Pohlheim H. and Fonseca C. (1993). *Genetic Algorithm Toolbox for use with MATLAB.* Dept. of Automatic Control and Sys. Eng., University of Sheffield, ver. 1.2, 1993.

Chu P.L. (1985). Quadrature mirror filter design for an arbitrary number of equal bandwidth channels. *IEEE Transactions on Acoustics, Speech and Signal Processing*. vol. **33**(1), pp.203-218.

Ciloglu T. (2002). An efficient local search method by gradient information for discrete coefficient FIR filter design. *Journal of Signal Processing*, Elsvier Science BV. vol. **82**(10), pp. 1337-1350.

Cox R.V. (1986). The design of uniformly and nonuniformly spaced pseudo QMF. *IEEE Transactions on Acoustics, Speech, Signal Processing*. vol. **34**, pp. 1090-1096.

Crochiere R.E. (1981) Sub-Band coding. *Bell System Journal*. vol. **60**(7), pp. 1633-1653.

Dempster A.G. and Macleod M.D (1994). Multiplier blocks and the complexity of IIR structures. *IEE Electronics letters*. vol. **30**, pp. 1841-1842.

Fettweis A. (1974). Wave digital lattice filters. *International Journal on Circuit theory application.* vol. **2,** pp 203-211.

Fliege N.J. (1994). *Multirate digital signal processing.* Chichester, U.K: John Wiley and Sons Ltd.

Fonseca C. and Fleming P.J. (1998). Multiobjective optimization and multiple constraint handling with evolutionary algorithms – Part I: A unified formulation. *IEEE Transactions on Systems, Man., and Cybernetics.* vol. **28**(1), Jan 1998.

Goh C.K. and Lim Y.C. (1998). An efficient algorithm for the design of weighted minimax M-channel cosine-modulated filter banks. *IEEE Transactions on Signal Processing.* vol. **46**(5), pp.1426-I430.

Goldberg D.E. (1989). Genetic algorithms in search optimisation and machine learning. New York: Addison Wesley.

Gray A.H. and Markel J.D.(1973). Digital Lattice and Ladder filter synthesis. *IEEE Transactions on audio and electroacoustics.* vol. **21**, pp 491-500.

Harris P.S. and Ifeachor E.C. (1995). Automating IIR filter design. *Proceedings of IEE International conference on Genetic Algorithms in Engineering Systems: Innovations and Applications (GALESIA, 1995),* pp.271-275.

Harris S.P. and Ifeachor E.C. (1998). Automatic design of frequency sampling filters by hybrid genetic algorithm techniques. *IEEE Transactions on Signal Processing.* vol. **46**(12), pp. 3304-3314.

Hoang P-Q and Vaidyanathan P.P (1989). Non-uniform multirate filter banks: theory and design. *Proceedings of IEEE International Symposium on Circuits and Systems,* Portland, OR, pp.371-374.

Holland J. (1975). *Adaptation in Natural and Artificial Systems.* Ann Arbor: University of Michigan Press.

Ifeachor E.C. and Jervis B.W. (1993). *Digital Signal Processing: A practical approach.* Wokingham, England: Addison-Wesley publishing.

Jackson L.B. (1989). *Digital filters and signal processing.* Boston: Kluwer.

Jayant N.S. and Noll P. (1984). *Digital coding of waveforms.* Englewood Cliffs: Prentice-Hall.

Jeong-jin L. and Byeong G.L. (1995). A design of nonuniform cosine modulated filter banks. *IEEE Transactions on Circuits and Systems.* vol. **42**(11), pp. 732-737.

Kaiser J.F. (1965). Some practical considerations in the realization of linear digital filters. *Proceedings of the 3rd Annual Allerton conference on Circuit System Theory.* pp 621-633.

Kaiser J.F. (1966). *Digital filters* - chapter 7 in: Systems analysis by digital computer. New York: Wiley 1966.

Kodek D.M. and Steiglitz K. (1981). Comparison of optimal and local search methods for designing finite wordlength FIR digital filters. *IEEE Transactions on Circuits and Systems.* vol. **28**(1), pp.28-32.

Kodek D.M. (1980). Design of optimal finite wordlength FIR digital filters using integer programming techniques. *IEEE Transactions on Acoustics, Speech and Signal Processing.* vol. **28**(3), pp.304-308.

Kovacevic J and Vetterli M. (1993). Perfect reconstruction filter banks with rational sampling factors. *IEEE Transactions on Signal Processing.* vol.**41**, pp. 2047-2064.

Kwong S. and Chau C.W. (1997). Analysis of parallel genetic algorithms on HMM based speech recognition system. *IEEE Transactions on Consumer Electronics.* vol. **43**(4), pp. 1229-1233

Lee J.-H. and Niu I.-C. (2001). Minimax design of two-channel IIR QMF banks with arbitrary group delay. *IEE Proceedings on Vision, Image and Signal Process.* vol. **148**(6), pp. 384-390.

Lin Y.P. and Vaidyanathan P.P. (1995). Linear-phase cosine-modulated maximally decimated filter banks with perfect reconstruction. *IEEE Transactions of Signal Processing.* vol. **43**(11), pp.2525-2539.

Lu H.C. and Tzeng S.T. (2000). Design of arbitrary FIR log filters by genetic algorithm approach. *Journal of Signal Processing,* Elsevier Science BV. vol. **80**(3), pp. 497-505.

McClellan J.H. (1973). The design of two-dimensional digital filters by transformations. *Proceedings of the 7th Princeton Conf. On Info. Sciences and Syst.* pp. 247-251.

Mehr A.S. and Chen T.W. (1999). Optimal design of nonuniform multirate filter banks. *Journal on Circuits Systems and Signal Processing*: Birkhauser Boston. vol. **18**(5), pp.505-521.

Mehr A.S. and Chen T.W. (2000). Design of nonuniform multirate filter banks by semidefinite programming. *IEEE Transactions on Circuits and Systems II.* vol. **47**(11), pp.1311-1314.

MELP standard draft (1998). *Specifications for the Analog to Digital conversion of voice by 2,400 bit/second mixed excitation linear prediction.* US Federal Standard for MELP coder, 1998.

Mitra S.K. (1998). *Digital Signal Processing – a computer based approach.* New York: McGraw-Hill.

Muhlenbein H. and Schlierkamp-Voosen D. (1993). Predictive models for the breeder genetic algorithm. *Journal of Evolutionary computation.* vol. **1**(1), pp.25-49.

Nayebi K., Barwell T.P. and Smith M.J.T. (1993). Nonuniform filter banks: a reconstruction and design theory. *IEEE Transactions on Signal Processing.* vol. **41** pp.1114-1127.

Nelder J.A. and Mead R. (1965). A Simplex method for function minimization. *Computer Journal.* vol. **7**, pp.308-313.

Nguyen T.Q. (1992). A class of generalised cosine-modulated filter bank. *IEEE Proceedings of Int. Symp., Circuits and Sys.* pp. 943-946, San Diego, CA, May 1992.

Oates M., Corne D. and Turton B.C.H. (1999). The effects of Selection Pressure on parameter choice in evolutionary search. *Genetic and Evolutionary Computation Conference, Orlando, Florida*. late breaking paper pp. 198-203, July 1999.

Oppenheim A.V and Shaffer R.W. (1989). *Discrete-time Signal Processing.* Englewood Cliffs, N.J: Prentice-Hall.

Parks T.W. and Burrus C.S. (1987). *Digital Filter Design*. New York: John Wiley, 1987.

Phoong S.M., Kim C.W., Vaidyanathan P.P. and Ansari R. (1995). A new class of two-channel biorthogonal filter banks and wavelet bases. *IEEE Transaction on Signal Processing*. vol. **43**(3), pp.649-665.

Porat B. A. (1997). *Course in Digital Signal Processing*. New York:Wiley.

Press W. H., Flannery B. P., Teukolsky S. A., and Vetterling W. T. (1989). *Numerical Recipes in FORTRAN: The Art of Scientific Computing*, 2nd ed. Cambridge, England: Cambridge University Press.

Redmill D.W. and Bull D.R. (1997). Design of low complexity FIR filters using genetic algorithms and directed graphs. *Proceedings of IEE International conference on Genetic Algorithms in Engineering Systems: Innovations and Applications (GALESIA, 1997).* pp.168-173, September 1997.

Redmill D.W. and Bull D.R. (1998). Automated design of low complexity FIR filters. *IEEE International Symposium on Circuits and Systems*. No.5, pp.429-432.

Roberts R.A and Mullis C.T. (1987). *Digital Signal Processing*. Reading, MA: Addison-Weseley.

Rothweiler J.H. (1983). Polyphase quadrature filters, a new subband coding technique. *Proceedings of the IEEE Int. Conf. On ASSP*. pp.1980-1983, April 1983

Suckley D. (1991). Genetic algorithm in the design of FIR filters. *IEE Proceedings on circuits, devices and systems*. vol. **138**(2), pp 234-238.

Tang K-s, Man K-f, Kwong S. and Liu Z-f, (1998). Design and optimization of IIR filter structure using hierarchical genetic algorithms. *IEEE Transactions of Industrial Electronics*. vol. **45**(3), pp.481-487.

Tay D.B.H. and Kingsbury N.G. (1993). Flexible design of multidimensional perfect reconstruction FIR 2-band filters using transformations of variables. *IEEE Transactions on Image Processing*. vol. **2**(4), pp. 466-480.

Tay D.B.H. and Kingsbury N.G. (1996). Design of 2-D perfect reconstruction filter banks using transformations of variables: IIR case. *IEEE Transactions on Circuits and Systems -II Analog and Dig. Sig. Proc.* vol. **43**(3), pp. 274-279.

Tay D.B.H. (1998). Design of causal stable IIR perfect reconstruction filters banks using transformation of variables. *IEE Proceedings on Vision, Image and Signal Process.* vol. **145**(4), pp. 287-292.

[Texas Instruments, 1991] TMS320 Fixed point DSP assembly language tools - user's guide, Texas Instruments, 1991.

[Texas Instruments, 1994] TMS320C5X C Source debugger, Texas Instruments, 1994.

[Texas Instruments, 1996] TMS320C5x DSP starter kit – user's guide, Texas Instruments, 1996.

[TMS 320C5x DSK, 1997] TMS 320C5x DSK Applications guide and starter kit, Texas Instruments Inc., 1997

Vaidyanathan P.P, Regalia P.A. and Mitra S.K. (1987). Design of doubly-complementary IIR digital filters using a single complex allpass filter, with multirate applications. *IEEE Transactions on Circuits and Systems*. vol.**34**, pp 378-389.

Vaidyanathan P.P. (1987, a). Theory and design of M-channel maximally decimated quadrature mirror filters with arbitrary M, having perfect reconstruction property. *IEEE Trans. on Acoustics, Speech and Signal Processing*. vol. **35**, pp. 476-492.

Vaidyanathan P.P. (1987, b). Quadrature mirror filter banks, M-band extensions and perfect-reconstruction techniques. *IEEE Acoustics, Speech and Signal Processing* ASSP magazine. vol. 4, pp. 4-20.

Vaidyanathan P.P. (1990). Mutirate digital filters, filter banks, polyphase networks and applications: a tutorial. *IEEE Proceedings*. vol. **78,** pp. 56-93.

Vaidyanathan P.P. (1993). *Multirate systems and filter banks*. Prentice-Hall.

Vetterli M. and Kovacevic J. (1995). *Wavelets and sub-band coding*. Prentice Hall.

Wada S. (1995). Design of non-uniform division multirate FIR filter banks. *IEEE Transactions on Circuits and Systems – II.* vol. **42**(2), pp.115-121.

Wade G., Van-Eetvelt P. and Darwen H. (1990). Synthesis of efficient low-order FIR filters from primitive sections. *IEE Proceedings part G.* vol. **137**, pp.367-372.

Whitley D. (1989). The GENITOR algorithm and selection pressure: why rank-based allocation or reproductive trials is best. *Proceedings of the International conference on genetic algorithms (ICGA-3)*. pp.116-121. Morgan Kaufmann Publishers.

Wilson P.B. and Macleod M.D. (1993). Low implementation cost IIR digital filter design using genetic algorithms. *Proceedings of the workshop on "Natural algorithms in signal processing"*. vol.1, pp.4/1-4/8, Chelmsford, U.K., Nov. 1993.

Wright A.H. (1991). Genetic algorithms for real parameter optimization in *Foundations of Genetic Algorithms,* J.E.Rawlins (Ed), Morgan Kaufmann, pp.205-218.

Xu D.J. and Daley M.L. (1995). Design of optimal digital filter using a parallel genetic algorithm. *IEEE Transactions on Circuits and Systems – II.* vol. **42**(10), pp.673-675.

Yu Y.J. and Lim Y.C. (2002). A novel genetic algorithm for the design of a signed power-of-two coefficient quadrature mirror filter lattice filter bank. *Journal of Circuits Systems and Signal Processing*. Birkhauser Boston Inc. vol. **21**(3), pp.263-276.

Zadeh L.A. (1963). Optimality and nonscalar-valued performance criteria. *IEEE Trans. Automat. Contr.* vol. **8**, 1963.

Zhu W.-P., Ahmad M.O. AND Swamy M.N.S. (1998). An efficient approach for the design of nearly perfect-reconstruction QMF banks. *IEEE Transactions on Circuits and Systems – II: Analog and Digital Sig. Proc,* vol. **45**(8), Aug 1998.

Appendices

Appendix A

Published Papers

A1 Published Papers

This appendix contains abstracts of papers published during the course of this study.

A1.1 Optimisation of finite word length FIR and IIR digital filters through genetic algorithms

Published in the proceedings of the IEE sponsored International Conference on Communication, Computer & Power, Muscat, Oman, pp. 183-187, December 7-10, 1998.

This paper describes a MATLAB based genetic algorithm (GA) method for optimisation of frequency response of finite and infinite impulse response (FIR/IIR) digital filters under the constraint of finite word length (FWL) coefficients. A quantifiable approach is followed for comparing the results. The constraints for optimisation of FIR and different structures for IIR filters is considered and subsequently applied to the GA optimisation procedure.

A1.2 A two stage genetic algorithm for optimisation of causal IIR perfect reconstruction multirate filter banks

Published in the proceedings of the IEEE and IEE sponsored Congress on Evolutionary Computation (CEC 99), Washington D.C., USA, pp 897-903, July 6-9, 1999.

This paper describes a procedure for the optimisation of two channel perfect reconstruction filter banks using causal infinite impulse response (IIR) digital filters through the transformation of variables method. A two stage genetic algorithm is used for the optimisation procedure where the first stage GA searches for near optimal values of the transformation function variables. The second stage GA optimises the filter coefficients for finite word-length effects for real time implementation of causal IIR filters. Some examples are presented and their results included to show the efficacy of this method for real time implementation.

A1.3 Comparative study for optimisation of causal IIR perfect reconstruction filter banks

Published in the proceedings of the IEEE and IEE sponsored Congress on Evolutionary Computation (CEC 00), San Diego, USA, pp 974-977, July 16-19, 2000.

This paper considers a comparative study for optimisation of causal infinite impulse response (IIR) filters with applications to perfect reconstruction quadrature mirror filter (QMF) banks. A hybrid procedure is applied where a constrained genetic algorithm is first used to search the objective function landscape for a promising valley. Sub-optimal filter parameters are then further optimised using four different methods. These are; new GA based 'creep code', gradient based constrained Sequential Quadratic Programming the Quasi-Newton method and a non-gradient based downhill Simplex method. Finally, a comparison is drawn between each of the four methods of optimisation

A1.4 Optimisation of a class of non-uniform multirate filter banks - a genetic algorithms approach

Presented at the IEEE sponsored 3rd International Conference on Information, Communications and Signal Processing, (ICICS, 2001), Singapore, October 15-18, 2001.

This paper considers the design and optimisation issues of a class of non-uniform filter (NUF) banks. Standard optimisation method based on down hill Simplex algorithm is applied after an initial search over a wider landscape using genetic algorithms (GAs). This hybrid approach helps in locating an optimal minima point. The optimisation process is applied to the transfer function of the entire network to minimise the amplitude and aliasing distortions. The design method of NUF banks is based on using multiple low pass prototype filters that are transformed by sine or cosine multiplication to the appropriate filter banks. Maximally integer decimated structures are considered using FIR filters.

Appendix B

GA code for FIR filter

B1 GA code for FIR filter

B1.1 GA code for finite word length coefficient optimisation of an FIR digital filter

```
% fir_ga.m
%
% [bo]=fir_ga(b,n,BASE,PRSZ)
%
% GA optimisation of a FIR filter
% Returns GA optimised vector bo given vector b of an FIR
% filter and the quantisation factor (Ex. for Q15 format n=15)
% BASE is the maximum peak variance within the population
% For PRSZ=1 the GA will preserve zero patterns within the filter
% For PRSZ=0 the GA will ignore zero patterns within the filter
%
% version 2: created: 3rd Jan 2002
%
% filter 1: A15/5 - specifications

fp = 0.4;
fs = 0.5;
f = [0 fp fs 1];  % frequency band edges
m = [1 1 0 0];    % desired magnitude response
ne = 14;          % filter order
bok = [0 1 1 0 -2 1 5 7 5 1 -2 0 1 1 0];
b = remez(ne,f,m);
nbits=5;
bok1=bok(1:(ne/2));
bok=[bok fliplr(bok1)];
b = remez(ne,f,m,w);

PRSZ=0;
BASE=1;
n=nbits-1;

% GA parameters

NIND=100;
MAXGEN=10;
GGAP=0.8;
INSR=0.8;
L=500;

[M,w]=freqz(b,1,L);
M=abs(M);

lb=length(b);
lc=ceil(lb/2);
coefs=[round((2^n)*b(1:lc))];

if 2*lc > lb
      coefs2=coefs(1:lc-1);
      coefsrnd=[coefs fliplr(coefs2)];
  else
      coefsrnd=[coefs fliplr(coefs)];
end
```

```
   coefsrnd=coefsrnd/2^n;

[R,w]=freqz(coefsrnd,1,L);
R=abs(R);

l=length(coefs);

if PRSZ==1
   for i=1:l
%     if b(i)==0          % preserve zeros in b(i)
      if coefs(i)==0      % preserve zeros in rounded coeffs.
         MSK(i)=0;
      else
         MSK(i)=1;
      end
   end
else
   MSK=ones(1,l);
end

% Built field descriptor
for i=1:l
   FieldDR(1,i)=coefs(i)-BASE-0.4999;
   FieldDR(2,i)=coefs(i)+BASE+0.4999;
end
FieldDR(1,:)=FieldDR(1,:).*MSK;
FieldDR(2,:)=FieldDR(2,:).*MSK;

warning off;
% Initialise population
Chrom=round(crtrp(NIND,FieldDR));

% Evaluate initial population
ObjV=fir_obj(Chrom,lb,lc,M,L,n,R,fp,fs);

[m,z]=max(ObjV);
Chrom(z,:)=coefs;

ObjV=fir_obj(Chrom,lb,lc,M,L,n,R,fp,fs);

gen=0;   %counter

% Generational loop
while gen < MAXGEN

   %Assign fitness values to entire population
   FitnV = ranking(ObjV);

   %Select individuals for breeding
   SelCh=select('sus', Chrom, FitnV, GGAP);

   %Recombine individuals (crossover)
   SelCh=recombin('recdis',SelCh);

   %Evaluate offspring, call objective function
   ObjVSel=fir_obj(SelCh,lb,lc,M,L,n,R,fp,fs);

   %Reinsert offspring into population
```

```
   [Chrom ObjV]=reins(Chrom,SelCh,1,[1 INSR],ObjV,ObjVSel);

   %Increment counter
   gen=gen+1;
   [m,z]=min(ObjV);
   OBJ=ObjV(z,1);
  end
warning on;
[m,z]=min(ObjV);
for i=1:lc
   bo(i)=Chrom(z,i);
end
for i=(lc+1):lb
   bo(i)=Chrom(z,lb+1-i);
end
bo;
bo=bo/(2^n);
[GAop,w]=freqz(bo,1,w);
GAop=abs(GAop);

% plot results here
GAp=GAop(1:(fp*L));
GAs=GAop((fs*L):L);

Mp=M(1:(fp*L));
Ms=M((fs*L):L);

Rp=R(1:(fp*L));
Rs=R((fs*L):L);

opkp=opk(1:(fp*L));
opks=opk((fs*L):L);

range=(((ne+2)/2):ne+1);
b_GA=(bo*2^n);
b_GA=b_GA(range)';
b_rnd=(coefsrnd*2^n);
b_rnd=b_rnd(range)';
b_IP=bok(range)';
b_GRIP=[b_GA b_rnd b_IP]'

max_err_D_GA=max(max(abs(1-GAp)),max(abs(GAs)));
max_err_D_RD=max(max(abs(1-Rp)),max(abs(Rs)));
max_err_D_IP=max(max(abs(1-opkp)),max(abs(opks)));
max_err_D_EX=max(max(abs(1-Mp)),max(abs(Ms)));

max_err_D_BD=(2^(-nbits))*(sqrt(((2*(ne+1))-1)/3))+ max_err_D_EX;

maxerr_D_GRIPEB=[max_err_D_GA max_err_D_RD max_err_D_IP max_err_D_EX
max_err_D_BD]

sum_err_D_GA=(sum((abs(1-GAp)).^2)+sum((abs(GAs)).^2));
sum_err_D_RD=(sum((abs(1-Rp)).^2)+sum((abs(Rs)).^2));
sum_err_D_IP=(sum((abs(1-opkp)).^2)+sum((abs(opks)).^2));
sum_err_D_EX=(sum((abs(1-Mp)).^2)+sum((abs(Ms)).^2));
sumerr_D_GRIPE=[sum_err_D_GA sum_err_D_RD sum_err_D_IP sum_err_D_EX]
% end of GA
```

```
% Optimisation function for FIR filter
% utility code for use with fir_ga.m optimisation

function f=fir_obj(Chrom,lb,lc,M,L,n,R,fp,fs);


S=size(Chrom);

for i=1:S(1);

for k=1:lc
   bb(k)=Chrom(i,k);
end

for k=(lc+1):lb
   bb(k)=Chrom(i,lb+1-k);
end

[GA,w]=freqz(bb/(2^n),1,L);
GA=abs(GA);
GAp=GA(1:(fp*L));
GAs=GA((fs*L):L);

% objective function
f(i,1)=(sum((abs(1-GAp)).^2)+sum((abs(GAs)).^2))+10*max((max(abs(1-
GAp))),max(abs(GAs)));

% alternative objective function
% f(i,1)=max((max(abs(1-GAp))),max(abs(GAs)));

end;
```

Appendix C

GA code and optimised results for IIR filters

C1 GA code and optimised results for direct form IIR filter

C1.1 GA code for finite word length coefficient optimisation of a direct form IIR digital filter

```
% iir_ga.m
%
% [bo,ao]=iir_ga(b,a,nbits,BASE)
%
% GA optimisation of IIR direct form filter
% Returns GA optimised vectors 'bo' and 'ao' given by vectors
% 'b' and 'a' of an IIR filter coefficients represented by
% finite number of bits given by 'nbits'.
% BASE is the peak variance within the population set.
%
% G.S.Baicher  UWCN  version 2: April 2002

% GA characteristics
GGAP = 0.8;     % generation gap
INSR = 1;       % insertion rate of offspring
MAXGEN = 20;    % maximum number of generations
Nind = 120;     % population size

% Digital Filter characteristics

nbits = 12;     % number of bits
BASE = 1;
n = 4;          % filter order
L = 500;        % number of frequency points
nl = n+1;       % filter order plus 1
maxnum = 2^nbits - 1;    % maximum number

% Filter design procedure
% could use several filter design functions such as
% butter, cheby, ellip, yulewalk etc.

% elliptical filter

Wn = 0.5;       % Cut-off frequency for low pass filter
Rp = 1;         % dBs of ripple in passband
Rs = 40;        % stop band attenuation dBs
[b,a] = ellip(n,Rp,Rs,Wn);
[h,w] = freqz(b,a,L);       % frequency response
h = abs(h);                 % magnitude response
p = angle(freqz(b,a,L));    % phase response
x = [b,a];
% Set the maximum and minimum values for coefficients
% lower bound 'vlb' and upper bound 'vub'
   if (any(x < 0))
% If there are negative coefficients then sign bit is used
   maxnum = floor(maxnum/2);
   vlb = -maxnum * ones(1, 2*nl);
   vub = maxnum * ones(1, 2*nl);
      else
% otherwise, all positive coefficients
   vlb = zeros(1,2*nl);
```

```
   vub = maxnum * ones(1, 2*n1);
end

% Set the biggest value equal to maxnum.
[m, mix] = max(abs(x)); % m = max value, mix = i_th element
factor =  maxnum/m;
x =  factor * x;        % Rescale other filter coefficients
xorig = x;

% filter with coefficients simply rounded
xr = round(x);
br = xr(1:n1);
ar = xr(n1+1:2*n1);
hr = abs(freqz(br,ar,L));
pr=angle(freqz(br,ar,L));

xmask = 1:2*n1;

% Remove the largest coefficient value from the list
% of values that can be changed

xmask([mix]) = [];

% random number added to coefficients + or - Base value
% except the largest, used in create population function
BaseV = 2*BASE;

% create new population

Chrom = iir_pop(Nind,xr,xmask,BaseV,n)';

Best = NaN*ones(MAXGEN+1,1);  % Reset Counter
gen = 0                       % generational counter

% work out the object function value for each chromosome
ObjVal = iir_objf(Chrom,n1,h,hr,p,x,L);

% best chromosome for minimum object function value
[Best(gen+1),ix] = min(ObjVal);
Best
xcbest = Chrom(:,ix);
xcbest1 = xcbest';

% start of generational loop
while gen < MAXGEN

   % assign fitness value to each individual in population
   FitnV = ranking(ObjVal);

   % select good individuals for breeding
   SelCh = select('sus',Chrom',FitnV,GGAP);

   % recombine selected individuals - crossover
   SelCh = recombin('recdis', SelCh, 1);

   % evaluate object function of offsprings
   ObjVOff = feval('iir_objf',SelCh',n1,h,hr,p,x,L);
```

```
   % reinsert good offsprings into current population
   [Chrom, ObjVal] = reins(Chrom', SelCh, 1, [1 INSR], ObjVal,
ObjVOff);

   Chrom = Chrom';

   %increment generational counter
   gen=gen+1

   % update display and remember the best individual to date
   [Best(gen+1,1),ix] = min(ObjVal);
   Best

   xcbest = Chrom(:,ix);
   xcbest1 = xcbest';
end

% results
% best individual performance
bo = xcbest1(1:n1);
ao = xcbest1(n1+1:2*n1);
ho = abs(freqz(bo,ao,L));
po = angle(freqz(bo,ao,L));
err_GA = sum(abs(h-ho))     % total GA-opt magnitude error
err_rnd = sum(abs(hr-h))    % total rounded coef magnitude error
% end of GA



% iir_objf.m
% Calculate object function value for GA code iir_ga.m
% G.S.Baicher  UWCN  version 2: April 2002

function ObjVal = iir_objf(Chrom,n1,h,hr,p,x,L);

% work out population parameters
[Nind,Nvar] = size(Chrom);

% start of loop for each individual
for irun = 1:Nvar;

% calculate coefficients ac and bc
 xc = Chrom(:,irun);
 bc = xc(1:n1);
 ac = xc(n1+1:2*n1);

% work out the roots of xc
g1 = [abs(roots(ac))];
g2 = [abs(roots(bc))];
% work out the absolute magnitude and angle
hc = abs(freqz(bc,ac,L));
pc = angle(freqz(bc,ac,L));

% stability criteria checked, if any root
% is > or = 1, then replace hc with large hc
if (any(g1 >= 1))
   hc = 100 + hc;
end
```

```
% minimum phase checked, if any root
% is > or = 1, then replace hc with large hc
if (any(g2>=1))
  hc = 100 + hc;
end
% work out magnitude and phase errors
mag_err = sum(abs(h-hc));
ph_err = sum(abs(p-pc));
m=floor(length(p)/2);

% work out a weighted object value between
% magnitude and phase errors
mag_er_rnd=sum(abs(h-hr));
ObjVal(irun,:) = mag_err + 0.001*ph_err;
end
```

```
% iir_pop.m
% create an initial real value population
% G.S.Baicher  UWCN  version 2: April 2002

function Chrom = iir_pop(Nind,xr,xmask,BaseV,n);

Nind=Nind-1;
% start a counter
count = 1;
a0=xr(n+2);
% continue while number of individuals in population
% is less than Nind
while count < Nind;

% add a random number to coefficients except the largest
 for k = 1:Nind;
  Fix = xmask(1);
  xnew = xr;

 for i = xmask;
  rndNu = (round(rand(1,1).*BaseV(ones(1,1))) - (BaseV/2))';
  xnew(i) = xnew(i) + rndNu;
  Fix = i;
xnew(n+2)=a0;
end

  xn1(k,:) = xnew;

count = count + 1;

end
end
Chrom = [xr
   xn1];
```

C1.2 GA optimised finite word length coefficients of direct form IIR digital filters

b,a – exact coefficient values; br, ar – rounded coefficients; bom, aom – GA optimised with minimal phase; bon, aon – GA optimised with non-minimal phase

filter	coefficients
Low-pass 4^{th} *order*	b = 0.1044 0.2702 0.3663 0.2702 0.1044 a = 1.0000 -0.6120 1.1131 -0.4946 0.2452
DF/LP4/5	br = 1 4 5 4 1 ar = 13 -8 15 -7 3 bom = 0 3 4 4 2 aom = 13 -8 15 -7 4 bon = 2 4 5 3 1 aon = 13 -8 15 -7 4
DF/LP4/8	br = 12 31 42 31 12 ar = 114 -70 127 -56 28 bo = 12 31 42 31 12 ao = 114 -69 127 -56 28 min. phase same as non-min. phase
DF/LP4/12	br = 192 497 674 497 192 ar = 1839 -1125 2047 -910 451 bo = 192 497 674 497 192 ao = 1839 -1125 2047 -910 451 min. phase same as non-min. phase
Low-pass 6^{th} *order*	b = 0.0757 0.1762 0.3244 0.3660 0.3244 0.1762 0.0757 a = 1.0000 -0.9901 2.2229 -1.6076 1.4668 -0.6326 0.2446
DF/LP6/5	br = 1 1 2 2 2 1 1 ar = 7 -7 15 -11 10 -4 2 bom = 0 2 1 2 2 1 1 aom = 7 -8 15 -10 10 -4 2
DF/LP6/8	br = 4 10 19 21 19 10 4 ar = 57 -57 127 -92 84 -36 14 bom = 5 11 19 21 18 10 4 aom = 57 -57 127 -91 84 -36 15 bon = 5 10 19 20 18 9 4 aon = 57 -57 127 -92 84 -36 14
DF/LP6/12	br = 70 162 299 337 299 162 70 ar = 921 -912 2047 -1480 1351 -583 225 bom = 71 162 299 336 299 162 71 aom = 921 -913 2047 -1480 1351 -583 226 bon = 70 162 299 337 299 162 70 aon= 921 -911 2047 -1480 1352 -583 226
Low-pass 8^{th} *order*	b = 0.0703 0.1534 0.3549 0.4694 0.5696 0.4694 0.3549 0.1534 0.0703 a = 1.0000 -1.1409 3.3229 -2.9368 3.8651 -2.4740 1.7875 -0.6780 0.2450
DF/LP8/5	br = 0 1 1 2 2 2 1 1 0 ar = 4 -4 13 -11 15 -10 7 -3 1 bo = 0 1 1 2 2 2 1 1 0 ao = 4 -4 13 -11 15 -10 7 -3 1 min. phase same as non-min. phase
DF/LP8/8	br = 2 5 12 15 19 15 12 5 2

	ar = 33 -37 109 -96 127 -81 59 -22 8 bom = 2 5 11 15 18 15 11 5 2 aom = 33 -38 109 -97 127 -81 59 -22 8 bon = 2 5 12 15 20 15 13 5 3 aon = 33 -38 109 -96 127 -80 60 -22 9
DF/LP8/12	br = 37 81 188 249 302 249 188 81 37 ar = 530 -604 1760 -1555 2047 -1310 947 -359 130 bom = 37 81 188 248 302 248 188 81 37 aom = 530 -604 1760 -1555 2047 -1310 947 -359 130 bon = 37 81 188 249 302 249 188 81 37 aon = 530 -604 1760 -1555 2047 -1310 947 -359 130

filter	**coefficients**
High-pass 4th order	b = 0.0619 -0.1103 0.1521 -0.1103 0.0619 a = 1.0000 1.5035 1.7320 0.9858 0.3146
DF/HP4/5	br = 1 -1 1 -1 1 ar = 9 13 15 9 3 bon = 1 -1 2 -1 1 aon = 9 12 15 8 3 bom = 0 -1 1 -1 0 aom = 9 14 15 9 3
DF/HP4/8	br = 5 -8 11 -8 5 ar = 73 110 127 72 23 bom = 6 -9 11 -7 4 aom = 73 110 127 72 23 bon = 5 -8 11 -8 4 aon = 73 110 127 72 23
DF/HP4/12	br = 73 -130 180 -130 73 ar = 1182 1777 2047 1165 372 bom = 74 -131 180 -129 72 aom = 1182 1778 2047 1165 371 bon = 74 -130 180 -131 73 aon = 1182 1776 2047 1165 372
High-pass 6th order	b = 0.0448 -0.0345 0.1062 -0.0714 0.1062 -0.0345 0.0448 a = 1.0000 2.4213 4.0615 4.1074 2.9524 1.3082 0.3193
DF/HP6/5	br = 0 0 0 0 0 0 0 ar = 4 9 15 15 11 5 1 bom = 0 0 0 0 0 0 0 aom = 4 10 15 15 11 5 1 bon = 1 1 1 1 1 0 0 aon = 4 10 15 15 11 5 1
DF/HP6/8	br = 1 -1 3 -2 3 -1 1 ar = 31 75 126 127 91 40 10 bom = 2 -1 4 -1 3 0 1 aom = 31 76 125 127 91 41 10 bon = 1 0 3 -1 4 -1 2 aon = 31 76 125 127 91 41 10
DF/HP6/12	br = 22 -17 53 -36 53 -17 22 ar = 498 1207 2024 2047 1471 652 159 bom = 23 -18 53 -36 52 -18 22 aom = 498 1207 2025 2047 1471 652 159

	bon = 22 -17 52 -36 53 -17 23 aon = 498 1207 2024 2047 1471 652 159
High-pass *8th order*	b = 0.0417 0.0006 0.1216 -0.0212 0.1571 -0.0212 0.1216 0.0006 0.0417 a = 1.0000 3.1620 6.8304 9.5157 10.0263 7.6017 4.2435 1.5524 0.3205
DF/HP8/5	None
DF/HP8/8	None
DF/HP8/12	br = 9 0 25 -4 32 -4 25 0 9 ar = 204 646 1395 1943 2047 1552 866 317 65 bom = 9 0 25 -4 32 -4 25 0 9 aom = 204 646 1395 1942 2047 1551 866 316 65 bon = 8 0 24 -4 32 -5 25 -1 9 aon = 204 646 1395 1942 2047 1552 867 317 65

filter	**coefficients**
Band-pass *4th order*	b = 0.2234 0.0000 -0.4300 -0.0000 0.2234 a = 1.0000 -0.0000 0.3491 -0.0000 0.3330
DF/BP4/5	br = 3 0 -6 0 3 ar = 15 0 5 0 5 bom = 4 0 -6 0 3 aom = 15 0 5 0 5 bon = 2 0 -6 0 4 aon = 15 0 6 0 5
DF/BP4/8	br = 28 0 -55 0 28 ar = 127 0 44 0 42 bom = 29 0 -55 0 28 aom = 127 0 44 0 42 bon = 29 0 -55 0 28 aon = 127 0 44 0 42
DF/BP4/12	br = 457 0 -880 0 457 ar = 2047 0 715 0 682 bom = 457 0 -880 0 457 aom = 2047 0 716 0 682 bon = 457 0 -880 0 457 aon = 2047 0 716 0 682
Band-pass *6th order*	b = 0.0982 0.0000 -0.2162 -0.0000 0.2162 0.0000 -0.0982 a = 1.0000 -0.0000 0.9531 -0.0000 0.8714 -0.0000 0.2895
DF/BP6/5	br = 1 0 -3 0 3 0 -1 ar = 15 0 14 0 13 0 4 bom = 2 0 -3 0 3 0 -1 aom = 15 0 15 0 13 0 4 bon = 2 0 -3 0 3 0 -1 aon = 15 0 15 0 13 0 4
DF/BP6/8	br = 12 0 -27 0 27 0 -12 ar = 127 0 121 0 111 0 37 bom = 12 0 -27 0 27 0 -12 aom = 127 0 122 0 110 0 36 bon = 13 0 -28 0 27 0 -12 aon = 127 0 121 0 111 0 37
DF/BP6/12	br = 201 0 -443 0 443 0 -201 ar = 2047 0 1951 0 1784 0 593

	bom = 201 0 -443 0 442 0 -200 aom = 2047 0 1952 0 1784 0 593 bon = 201 0 -443 0 442 0 -201 aon = 2047 0 1951 0 1784 0 593
Band-pass 8th order	b = 0.0619 0.0000 -0.1103 -0.0000 0.1521 0.0000 -0.1103 -0.0000 0.0619 a = 1.0000 -0.0000 1.5035 -0.0000 1.7320 -0.0000 0.9858 -0.0000 0.3146
DF/BP8/5	br = 1 0 -1 0 1 0 -1 0 1 ar = 9 0 13 0 15 0 9 0 3 bom = 1 0 -1 0 1 0 0 0 0 aom = 9 0 14 0 15 0 9 0 3 bon = 1 -1 -1 0 1 -1 -1 0 0 aon = 9 0 13 0 15 0 9 0 3
DF/BP8/8	br = 5 0 -8 0 11 0 -8 0 5 ar = 73 0 110 0 127 0 72 0 23 bom = 5 0 -8 0 11 0 -8 0 5 aom = 73 0 110 0 127 0 72 0 23 bon = 5 0 -8 0 11 0 -8 0 4 aon = 73 0 110 0 127 0 72 0 23
DF/BP8/12	br = 73 0 -130 0 180 0 -130 0 73 ar = 1182 0 1777 0 2047 0 1165 0 372 bom = 73 0 -130 0 180 0 -130 0 73 aom = 1182 0 1777 0 2047 0 1165 0 372 bon = 73 0 -130 0 180 0 -130 0 73 aon = 1182 0 1778 0 2047 0 1165 0 371

C2 GA code and optimised results for second order cascade form IIR filters

C2.1 GA code for finite word length coefficient optimisation of a second order cascade form IIR digital filter

```
% sos_ga.m
% soso=sos_ga(sosba,nbits,BASE)
% GA optimisation of IIR filter finite wordlength
% coefficients using biquad second order sections in a cascade
%
% Returns GA optimised vector 'soso' given by vector
% 'sosba' of an IIR filter coefficients represented by
% finite number of bits given by 'nbits'.
% BASE is the peak variance within the population set.
%
% G.S.Baicher  UWCN  version 2: April 2002


% GA characteristics

GGAP = .8;      % generation gap
INSR = 1;       % insertion rate of offspring
MAXGEN = 15;    % maximum number of generations
Nind = 120;     % population size

% Digital Filter characteristics
n = 8;          % filter order
L = 500;        % number of frequency points
nbits = 8;  % start with number of bits
maxnum = 2^nbits - 1;    % maximum number

% Filter design procedure

% elliptical filter

Wn = 0.5;       % Cut-off frequency for filter
Rp = 1;         % dBs of ripple in passband
Rs = 40;        % stop band attenuation dBs
[b,a] = ellip(n,Rp,Rs,Wn);
sos=tf2sos(b,a,'up','none');  % 'None' norm
%sos=tf2sos(b,a,'up','inf');  % 'Infinity' norm
sosba=sos;
[h,w] = freqz(b,a,L);          % frequency response
h = abs(h);                    % magnitude response
p = angle(freqz(b,a,L));       % phase response
x = [b,a];

% Set the maximum and minimum values for coefficients
% lower bound 'vlb' and upper bound 'vub'
if (any(any(sos < 0)))
% If there are negative coefficients then sign bit is used
   maxnum = floor(maxnum/2);
   vlb = -maxnum * ones(1, 6*(n/2));
   vub = maxnum * ones(1, 6*(n/2));
else
```

```
% otherwise, all positive coefficients
   vlb = zeros(1, 6*(n/2));
   vub = maxnum * ones(1, 6*(n/2));
end
% Set the biggest value equal to maxnum.
[m, mix] = max(max(abs(sos))); % m = max value, mix = i_th element
factor =  maxnum/m;
sos =  factor * sos;     % Rescale other filter coefficients
sosorig = sos;

% filter with coefficients simply rounded
sosr = round(sos);
sosrx = sosr/vub(1);
[br,ar]= sos2tf(sosrx);
hr = abs(freqz(br,ar,L));
sosra = [];
for i = 1:n/2            % for low and high pass filters
sosra = [sosra, sosr(i,:)];
end
[m, mix] = max(abs(sosra));

sosmask = [1:6*(n/2)];  % for low and high pass filters

% Remove the largest coefficient value
sosmask([mix]) = [];

% random number added to coefficients + or - (BaseV/2)
% except the largest, used in create population function
BaseV = 2;

% create new population
Chrom = sos_pop(Nind,sosra,sosmask,BaseV,maxnum)';
Best = NaN*ones(MAXGEN+1,1);  % Reset Counter
gen = 0;                      % generational counter

% work out the object function value for each chromosome
ObjVal = sos_objf(Chrom,h,p,sosra,L,maxnum);

% best chromosome for minimum object function value
[Best(gen+1),ix] = min(ObjVal);
Best;
xcbest = Chrom(:,ix);
xcbest1 = xcbest';

% start of generational loop
while gen < MAXGEN

   % assign fitness value to each individual in population
   FitnV = ranking(ObjVal);

   % select good individuals for breeding
   SelCh = select('sus',Chrom',FitnV,GGAP);

   % recombine selected individuals - crossover
   SelCh = recombin('recdis', SelCh, 1);

   % evaluate object function of offsprings
```

```
   ObjVOff = feval('sos_objf',SelCh',h,p,sosra,L,maxnum);

   % reinsert good offsprings into current population
   [Chrom, ObjVal] = reins(Chrom', SelCh, 1, [1 INSR], ObjVal,
ObjVOff);

Chrom = Chrom';

%increment generational counter
gen=gen+1
% update display and remember the best individual to date
[Best(gen+1,1),ix] = min(ObjVal);
Best
xcbest = Chrom(:,ix);
xcbest1 = xcbest';
end

% results: best individual performance
xc1 = xcbest1(1:6);
xc2 = xcbest1(7:12);
xc3 = xcbest1(13:18);
xc4 = xcbest1(19:24);
xcbest1 = [xc1;xc2;xc3;xc4];
soso=xcbest1;
xcbest1 = xcbest1./maxnum;
[bcbest1, acbest1] = sos2tf(xcbest1);
hcbest1 = abs(freqz(bcbest1,acbest1,L));
pcbest1 = angle(freqz(bcbest1,acbest1,L));
err_GA = sum(abs(hcbest1-h))
err_rnd= sum(abs(hr-h))
soso

% end of GA


% sos_objf.m
% Calculate object function value for
% GA  sos_ga.m
%
% G.S.Baicher  UWCN  version 2: April 2002

function ObjVal = sos_objf(Chrom,h,p,sosra,L,maxnum);

% work out population parameters
[Nvar, Nind] = size(Chrom);
% start of loop for each individual
for irun = 1:Nind
 xc = Chrom(:,irun);
 xc = xc';
% calculate roots for stability check
ac1 = xc(4:6);
 g1 = abs(roots(ac1));
ac2 = xc(10:12);
 g2 = abs(roots(ac2));
ac3 = xc(16:18);
 g3 = abs(roots(ac3));
ac4 = xc(22:24);
 g4 = abs(roots(ac4));
```

```
 g = [g1 g2 g3 g4];
xc1 = xc(1:6);
xc2 = xc(7:12);
xc3 = xc(13:18);
xc4 = xc(19:24);
xc = [xc1;xc2;xc3;xc4];
xc = xc./maxnum;
[bc, ac] = sos2tf(xc);
% work out the absolute magnitude and angle
  hc = abs(freqz(bc,ac,L));
  pc = angle(freqz(bc,ac,L));
% stability criteria checked
if any(any(g >= 1))
  hc = 100 + hc;
end
% work out magnitude and phase errors
mag_err = sum(abs(h-hc));
ph_err = sum(abs(p-pc));

% work out a weighted object value
ObjVal(irun,:) = mag_err + 0.001*ph_err;
end


% sos_pop.m
% create an initial real value population
%
% G.S.Baicher  UWCN  version 2: April 2002

function Chrom = sos_pop(Nind,sosra,sosmask,BaseV,maxnum);
Nind=Nind-1;
xr4=sosra(4);
% start a counter
count = 1;
% continue while number is less than Nind
while count < Nind;

% add a random number to coefficients except the largest
 for k = 1:Nind;
  Fix = sosmask(1);
  sosnew = sosra;
   for i = sosmask
  rndNu = (round(rand(1,1).*BaseV(ones(1,1))) - (BaseV/2))';
  sosnew(i) = sosnew(i) + rndNu;
   [m mix]=max(sosnew);
   if m>maxnum
     sosnew(mix)=maxnum;
  end
  Fix = i;
sosnew(4)=xr4; sosnew(10)=xr4; sosnew(16)=xr4; sosnew(22)=xr4;
end
  sosn1(k,:) = sosnew;
  count = count + 1;
end
end
Chrom = [sosra
   sosn1];
% end
```

C2.2 GA optimised finite word length coefficients of second order cascade form IIR digital filters

sosba – exact coefficients; sosr – rounded coefficients; soso – GA optimised
IN – 'Infinity' norm and NN – 'None' norm

filter	coefficients
Low-pass 4th order	sosba = 0.1011 0.1720 0.1011 1.0000 -0.6106 0.3029 1.0332 0.9153 1.0332 1.0000 -0.0013 0.8094
IN/SOS/LP4/5	sosr = 1 2 1 15 -9 4 15 13 15 15 0 12 soso = 2 3 1 15 -9 4 15 12 14 15 0 12
IN/SOS/LP4/8	sosr = 12 21 12 123 -75 37 127 113 127 123 0 99 soso = 13 21 12 123 -76 38 127 113 127 123 0 100
IN/SOS/LP4/12	sosr = 200 341 200 1981 -1210 600 2047 1813 2047 1981 -3 1604 soso = 200 341 200 1981 -1210 600 2047 1812 2046 1981 -3 1603
Low-pass 6th order	sosba = 0.1083 0.1723 0.1083 1.0000 -0.7713 0.3353 0.1674 0.0875 0.1674 1.0000 -0.2183 0.7651 4.1753 0.8969 4.1753 1.0000 -0.0006 0.9535
IN/SOS/LP6/5	sosr = 0 1 0 4 -3 1 1 0 1 4 -1 3 15 3 15 4 0 3 soso = 1 1 1 4 -3 1 1 0 0 4 0 3 15 2 14 4 0 3
IN/SOS/LP6/8	sosr = 3 5 3 30 -23 10 5 3 5 30 -7 23 127 27 127 30 0 29 soso = 4 5 2 30 -24 10 4 3 5 30 -7 22 127 28 127 30 0 28
IN/SOS/LP6/12	sosr = 53 84 53 490 -378 164 82 43 82 490 -107 375 2047 440 2047 490 0 467 soso = 53 84 53 490 -377 164 82 43 83 490 -107 375 2047 439 2047 490 0 467
Low-pass 8th order	sosba =

	0.1039 0.1624 0.1039 1.0000 -0.8081 0.3445 0.2191 0.0985 0.2191 1.0000 -0.2777 0.7639 0.1698 0.0206 0.1698 1.0000 -0.0550 0.9410 18.1744 0.8913 18.1744 1.0000 -0.0001 0.9892
IN/SOS/LP8/5	None
IN/SOS/LP8/8	sosr = 1 1 1 7 -6 2 2 1 2 7 -2 5 1 0 1 7 0 7 127 6 127 7 0 7 soso = 1 2 1 7 -5 2 1 1 1 7 -1 5 1 0 1 7 -1 6 127 6 127 7 0 6
IN/SOS/LP8/12	sosr = 12 18 12 113 -91 39 25 11 25 113 -31 86 19 2 19 113 -6 106 2047 100 2047 113 0 111 soso = 12 19 11 113 -90 38 25 12 25 113 -31 86 19 2 19 113 -6 106 2047 101 2047 113 0 112

filter	**coefficients**
High-pass *4th order*	sosba = 0.0619 -0.0911 0.0619 1.0000 0.9406 0.3846 1.0000 -0.3104 1.0000 1.0000 0.5630 0.8179
NN/SOS/HP4/5	sosr = 1 -1 1 15 14 6 15 -5 15 15 8 12 soso = 1 -1 1 15 15 6 14 -6 14 15 9 13
NN/SOS/HP4/8	sosr = 8 -12 8 127 119 49 127 -39 127 127 71 104 soso = 8 -12 8 127 119 49 126 -38 126 127 70 103
NN/SOS/HP4/12	sosr = 127 -186 127 2047 1925 787 2047 -635 2047 2047 1152 1674 soso = 127 -186 127 2047 1926 788 2047 -635 2047 2047 1153 1675
High-pass *6th order*	sosba = 0.0448 -0.0578 0.0448 1.0000 1.0735 0.4268 1.0000 0.1034 1.0000 1.0000 0.7429 0.7827 1.0000 0.4171 1.0000 1.0000 0.6049 0.9557
NN/SOS/HP6/5	sosr =

	1 -1 1 14 15 6 14 1 14 14 10 11 14 6 14 14 8 13 soso = 1 -1 0 14 15 6 14 1 14 14 10 11 14 7 15 14 8 13
NN/SOS/HP6/8	sosr = 5 -7 5 118 127 50 118 12 118 118 88 93 118 49 118 118 72 113 soso = 5 -7 5 118 127 50 117 11 118 118 89 92 117 49 117 118 72 113
NN/SOS/HP6/12	sosr = 85 -110 85 1907 2047 814 1907 197 1907 1907 1417 1493 1907 795 1907 1907 1153 1823 soso = 85 -110 86 1907 2047 814 1907 197 1907 1907 1417 1493 1908 795 1907 1907 1153 1823
High-pass *8th order*	sosba = 0.0417 -0.0520 0.0417 1.0000 1.1035 0.4378 1.0000 0.1808 1.0000 1.0000 0.7933 0.7835 1.0000 0.5062 1.0000 1.0000 0.6502 0.9442 1.0000 0.5733 1.0000 1.0000 0.6150 0.9898
NN/SOS/HP8/5	sosr = 1 -1 1 14 15 6 14 2 14 14 11 11 14 7 14 14 9 13 14 8 14 14 8 13 soso = 0 -1 1 14 15 6 14 1 14 14 10 11 15 8 15 14 9 13 15 9 13 14 7 12
NN/SOS/HP8/8	sosr = 5 -6 5 115 127 50 115 21 115 115 91 90 115 58 115 115 75 109 115 66 115 115 71 114 soso = 5 -6 5 115 127 51 114 21 114 115 91 90 115 58 116 115 75 109 114 67 115 115 71 114
NN/SOS/HP8/12	sosr = 77 -96 77 1855 2047 812 1855 335 1855 1855 1472 1453 1855 939 1855 1855 1206 1752 1855 1064 1855 1855 1141 1836 soso =

	78 -96 77 1855 2047 812 1855 335 1854 1855 1472 1453 1855 939 1854 1855 1207 1751 1854 1063 1855 1855 1141 1836

filter	**coefficients**
Band-pass *4th order*	sosba = 0.2014 0.3990 0.2014 1.0000 0.8972 0.5771 1.1095 -2.1979 1.1095 1.0000 -0.8972 0.5771
IN/SOS/BP4/5	sosr = 1 3 1 7 6 4 8 -15 8 7 -6 4 soso = 2 3 1 7 5 3 8 -15 8 7 -7 4
IN/SOS/BP4/8	sosr = 12 23 12 58 52 33 64 -127 64 58 -52 33 soso = 12 23 11 58 51 33 64 -127 64 58 -52 33
IN/SOS/BP4/12	sosr = 188 372 188 931 836 537 1033 -2047 1033 931 -836 537 soso = 189 372 187 931 836 537 1034 -2047 1034 931 -836 537
Band-pass *6th order*	sosba = 0.0935 -0.0000 -0.0935 1.0000 -0.0000 0.4487 0.7120 -1.2742 0.7120 1.0000 -1.0497 0.8032 1.4739 2.6376 1.4739 1.0000 1.0497 0.8032
IN/SOS/BP6/5	sosr = 1 0 -1 6 0 3 4 -7 4 6 -6 5 8 15 8 6 6 5 soso = 1 0 0 6 0 4 3 -6 3 6 -6 5 8 15 7 6 6 5
IN/SOS/BP6/8	sosr = 5 0 -5 48 0 22 34 -61 34 48 -51 39 71 127 71 48 51 39 soso = 4 -1 -5 48 0 22 34 -61 35 48 -51 39 71 127 72 48 51 39
IN/SOS/BP6/12	sosr = 73 0 -73 776 0 348

	553 -989 553 776 -815 623 1144 2047 1144 776 815 623 soso = 73 0 -72 776 0 348 552 -989 553 776 -814 623 1143 2047 1144 776 814 623
Band-pass *8th order*	sosba = 0.1622 0.3021 0.1622 1.0000 0.5475 0.6202 0.1735 -0.3233 0.1735 1.0000 -0.5475 0.6202 0.8287 -1.2597 0.8287 1.0000 -1.1162 0.9044 2.6559 4.0370 2.6559 1.0000 1.1162 0.9044
IN/SOS/BP8/5	sosr = 1 1 1 4 2 2 1 -1 1 4 -2 2 3 -5 3 4 -4 3 10 15 10 4 4 3 soso = 0 1 1 4 3 2 0 -1 1 4 -2 2 3 -5 3 4 -4 3 10 15 10 4 4 3
IN/SOS/BP8/8	sosr = 5 10 5 31 17 20 5 -10 5 31 -17 20 26 -40 26 31 -35 28 84 127 84 31 35 28 soso = 4 9 6 31 17 20 6 -10 5 31 -18 20 27 -39 25 31 -35 28 84 127 84 31 35 28
IN/SOS/BP8/12	sosr = 82 153 82 507 278 314 88 -164 88 507 -278 314 420 -639 420 507 -566 459 1347 2047 1347 507 566 459 soso = 82 153 82 507 277 314 88 -164 88 507 -278 314 420 -640 421 507 -566 458 1348 2047 1348 507 566 459

filter	**coefficients**
Low-pass 4th order	sosba = 0.1044 0.1777 0.1044 1.0000 -0.6106 0.3029 1.0000 0.8859 1.0000 1.0000 -0.0013 0.8094
NN/SOS/LP4/5	sosr = 2 3 2 15 -9 5 15 13 15 15 0 12 soso = 2 3 1 15 -9 4 14 12 15 15 0 12

NN/SOS/LP4/8	sosr = 13 23 13 127 -78 38 127 113 127 127 0 103 soso = 14 22 13 127 -78 39 126 114 127 127 0 103
NN/SOS/LP4/12	sosr = 214 364 214 2047 -1250 620 2047 1813 2047 2047 -3 1657 soso = 214 364 213 2047 -1250 620 2046 1813 2048 2047 -3 1656
Low-pass 6th order	sosba = 0.0757 0.1204 0.0757 1.0000 -0.7713 0.3353 1.0000 0.5230 1.0000 1.0000 -0.2183 0.7651 1.0000 0.2148 1.0000 1.0000 -0.0006 0.9535
NN/SOS/LP6/5	sosr = 1 2 1 15 -12 5 15 8 15 15 -3 11 15 3 15 15 0 14 soso = 1 2 1 15 -11 5 15 9 16 15 -2 11 15 3 16 15 0 14
NN/SOS/LP6/8	sosr = 10 15 10 127 -98 43 127 66 127 127 -28 97 127 27 127 127 0 121 soso = 10 15 9 127 -99 43 126 67 128 127 -28 97 127 28 127 127 0 121
NN/SOS/LP6/12	sosr = 155 246 155 2047 -1579 686 2047 1071 2047 2047 -447 1566 2047 440 2047 2047 -1 1952 soso = 155 246 155 2047 -1578 686 2047 1072 2047 2047 -447 1566 2047 440 2047 2047 -1 1952
Low-pass 8th order	sosba = 0.0703 0.1098 0.0703 1.0000 -0.8081 0.3445 1.0000 0.4498 1.0000 1.0000 -0.2777 0.7639 1.0000 0.1213 1.0000 1.0000 -0.0550 0.9410 1.0000 0.0490 1.0000 1.0000 -0.0001 0.9892
NN/SOS/LP8/5	sosr = 1 2 1 15 -12 5 15 7 15 15 -4 11 15 2 15 15 -1 14 15 1 15 15 0 15 soso = 1 2 1 15 -12 5 15 7 15 15 -3 11

	14 3 14 15 -1 14 15 1 14 15 1 14
NN/SOS/LP8/8	sosr = 9 14 9 127 -103 44 127 57 127 127 -35 97 127 15 127 127 -7 120 127 6 127 127 0 126 soso = 9 14 9 127 -103 44 127 57 126 127 -36 97 127 15 126 127 -7 120 127 6 127 127 0 126
NN/SOS/LP8/12	sosr = 144 225 144 2047 -1654 705 2047 921 2047 2047 -569 1564 2047 248 2047 2047 -113 1926 2047 100 2047 2047 0 2025 soso = 144 225 144 2047 -1654 705 2047 920 2047 2047 -569 1564 2047 248 2046 2047 -113 1927 2046 100 2046 2047 0 2025

filter	**coefficients**
High-pass *4th order*	sosba = 0.0619 -0.0911 0.0619 1.0000 0.9406 0.3846 1.0000 -0.3104 1.0000 1.0000 0.5630 0.8179
NN/SOS/HP4/5	sosr = 1 -1 1 15 14 6 15 -5 15 15 8 12 soso = 1 -1 1 15 15 6 14 -6 14 15 9 13
NN/SOS/HP4/8	sosr = 8 -12 8 127 119 49 127 -39 127 127 71 104 soso = 8 -12 8 127 119 49 126 -38 126 127 70 103
NN/SOS/HP4/12	sosr = 127 -186 127 2047 1925 787 2047 -635 2047 2047 1152 1674 soso = 127 -186 127 2047 1926 788 2047 -635 2047 2047 1153 1675
High-pass *6th order*	sosba = 0.0448 -0.0578 0.0448 1.0000 1.0735 0.4268 1.0000 0.1034 1.0000 1.0000 0.7429 0.7827 1.0000 0.4171 1.0000 1.0000 0.6049 0.9557
NN/SOS/HP6/5	sosr = 1 -1 1 14 15 6

	14 1 14 14 10 11 14 6 14 14 8 13 soso = 1 -1 0 14 15 6 14 1 14 14 10 11 14 7 15 14 8 13
NN/SOS/HP6/8	sosr = 5 -7 5 118 127 50 118 12 118 118 88 93 118 49 118 118 72 113 soso = 5 -7 5 118 127 50 117 11 118 118 89 92 117 49 117 118 72 113
NN/SOS/HP6/12	sosr = 85 -110 85 1907 2047 814 1907 197 1907 1907 1417 1493 1907 795 1907 1907 1153 1823 soso = 85 -110 86 1907 2047 814 1907 197 1907 1907 1417 1493 1908 795 1907 1907 1153 1823
High-pass *8th order*	sosba = 0.0417 -0.0520 0.0417 1.0000 1.1035 0.4378 1.0000 0.1808 1.0000 1.0000 0.7933 0.7835 1.0000 0.5062 1.0000 1.0000 0.6502 0.9442 1.0000 0.5733 1.0000 1.0000 0.6150 0.9898
NN/SOS/HP8/5	sosr = 1 -1 1 14 15 6 14 2 14 14 11 11 14 7 14 14 9 13 14 8 14 14 8 13 soso = 0 -1 1 14 15 6 14 1 14 14 10 11 15 8 15 14 9 13 15 9 13 14 7 12
NN/SOS/HP8/8	sosr = 5 -6 5 115 127 50 115 21 115 115 91 90 115 58 115 115 75 109 115 66 115 115 71 114 soso = 5 -6 5 115 127 51 114 21 114 115 91 90 115 58 116 115 75 109 114 67 115 115 71 114
NN/SOS/HP8/12	sosr = 77 -96 77 1855 2047 812 1855 335 1855 1855 1472 1453 1855 939 1855 1855 1206 1752 1855 1064 1855 1855 1141 1836 soso = 78 -96 77 1855 2047 812

	1855 335 1854 1855 1472 1453 1855 939 1854 1855 1207 1751 1854 1063 1855 1855 1141 1836

filter	**coefficients**
Band-pass *4th order*	sosba = 0.2234 0.4426 0.2234 1.0000 0.8972 0.5771 1.0000 -1.9811 1.0000 1.0000 -0.8972 0.5771
NN/SOS/BP4/5	sosr = 2 3 2 8 7 4 8 -15 8 8 -7 4 soso = 2 4 2 8 7 4 8 -15 7 8 -7 4
NN/SOS/BP4/8	sosr = 14 28 14 64 58 37 64 -127 64 64 -58 37 soso = 14 28 14 64 57 37 64 -127 64 64 -57 37
NN/SOS/BP4/12	sosr = 231 457 231 1033 927 596 1033 -2047 1033 1033 -927 596 soso = 231 457 230 1033 926 596 1033 -2047 1033 1033 -927 596
Band-pass *6th order*	sosba = 0.0982 -0.0000 -0.0982 1.0000 -0.0000 0.4487 1.0000 -1.7896 1.0000 1.0000 -1.0497 0.8032 1.0000 1.7896 1.0000 1.0000 1.0497 0.8032
NN/SOS/BP6/5	sosr = 1 0 -1 8 0 4 8 -15 8 8 -9 7 8 15 8 8 9 7 soso = 1 0 -1 8 0 3 8 -15 9 8 -9 6 8 14 8 8 9 6
NN/SOS/BP6/8	sosr = 7 0 -7 71 0 32 71 -127 71 71 -74 57 71 127 71 71 74 57 soso = 7 0 -7 71 1 32 71 -127 71 71 -74 57 71 127 72 71 75 57
NN/SOS/BP6/12	sosr = 112 0 -112 1144 0 513 1144 -2047 1144 1144 -1201 919 1144 2047 1144 1144 1201 919 soso =

	112 1 -113 1144 1 513 1144 -2047 1144 1144 -1201 919 1145 2047 1144 1144 1202 919
Band-pass *8th order*	sosba = 0.0619 0.1154 0.0619 1.0000 0.5475 0.6202 1.0000 -1.8630 1.0000 1.0000 -0.5475 0.6202 1.0000 -1.5200 1.0000 1.0000 -1.1162 0.9044 1.0000 1.5200 1.0000 1.0000 1.1162 0.9044
NN/SOS/BP8/5	sosr = 0 1 0 8 4 5 8 -15 8 8 -4 5 8 -12 8 8 -9 7 8 12 8 8 9 7 soso = 0 1 1 8 4 4 8 -15 8 8 -5 5 7 -11 8 8 -9 7 7 11 7 8 9 6
NN/SOS/BP8/8	sosr = 4 8 4 68 37 42 68 -127 68 68 -37 42 68 -104 68 68 -76 62 68 104 68 68 76 62 soso = 4 8 5 68 37 42 69 -127 69 68 -38 42 68 -103 68 68 -76 61 68 103 68 68 76 62
NN/SOS/BP8/12	sosr = 68 127 68 1099 602 681 1099 -2047 1099 1099 -602 681 1099 -1670 1099 1099 -1226 994 1099 1670 1099 1099 1226 994 soso = 68 127 68 1099 601 681 1099 -2047 1099 1099 -602 681 1099 -1670 1098 1099 -1227 994 1100 1670 1099 1099 1227 994

Appendix D

GA and hybrid optimisation codes for a QMF bank using IIR filters

D1 GA optimisation codes for a QMF bank using IIR filters

D1.1 GA optimisation of transformation function parameter values including the 'creep' code

```
% Main Code

% tv_ga
%
% GA optimisation of coefficients c,d,r,phi,p,psi
%
% G.S.Baicher  ver:2  Sept 1999

NIND=200;
MAXGEN=100;
GGAP=0.9;
INSR=0.7;
MutRate=0.2;
SWOV=10;     %Switchover operator
SWOVT=20;    %Switchover transition

b=0.7; n=25; wL=0.43*pi; L=100;
OBJ=0; clear j;
w=linspace(0,wL,L);
z=exp(j*w);
Qw=((pi/2)^(1-n))*w.^n;
Mi=cos(Qw)-j*b*sin(Qw);
W=1:L;
%W=fliplr(W); % Weighting - positive(%), negative( )

% Built field descriptor
FieldDR=[-1 -1 -2 -1 -2;1 1 2 1 2];

% Initialise population
Chrom=crtrp(NIND,FieldDR);

% Evaluate initial population
ObjV=tvgaobj(Chrom,z,Mi,b,W);

gen=0;   %counter
% Generational loop
while gen < MAXGEN

   %Assign fitness values to entire population
   FitnV = ranking(ObjV);

   %Select individuals for breeding
   SelCh=select('sus', Chrom, FitnV, GGAP);

   %Recombine individuals (crossover)
   SelCh=recombin('recdis',SelCh);

   %Apply mutation
   SelCh=mutbga(SelCh,FieldDR,MutRate);

   %Evaluate offspring, call objective function
```

```
   ObjVSel=tvgaobj(SelCh,z,Mi,b,W);

   %Reinsert offspring into population
   [Chrom ObjV]=reins(Chrom,SelCh,1,[1 INSR],ObjV,ObjVSel);

   %Increment counter
   gen=gen+1
   [m,n]=min(ObjV);
   ObjV(n,1)
   OBJ(gen)=ObjV(n,1);
   if gen > SWOVT
   disp('switchover')
   %[m,n]=min(ObjV);
   [e,u]=max(ObjV);
   for i=1:SWOV
      [e,u]=max(ObjV);
      Chrom(u,1)=Chrom(n,1)+((rand/50)-0.01);
      Chrom(u,2)=Chrom(n,2)+((rand/50)-0.01);
      Chrom(u,3)=Chrom(n,3)+((rand/50)-0.01);
      Chrom(u,4)=Chrom(n,4)+((rand/50)-0.01);
      Chrom(u,5)=Chrom(n,5)+((rand/50)-0.01);
      ObjV(u,1)=0;
   end
   ObjV=tvgaobj(Chrom,z,Mi,b,W);
   end

end

[m,n]=min(ObjV);
d=Chrom(n,1);
r=Chrom(n,2);
phi=Chrom(n,3);
p=Chrom(n,4);
psi=Chrom(n,5);
F1=((p^2)-2*p*cos(psi)+1)/((r^2)-2*r*cos(phi)+1);
F2=((r^2)+2*r*cos(phi)+1)/((p^2)+2*p*cos(psi)+1);
F=F1*F2;
c=(F*(1+d)-b*(1-d))/(F*(1+d)+b*(1-d));
f=m;
c,d,r,phi,p,psi,f
```

```
% Objective function for tv_ga.m

function f=tvgaobj(Chrom,z,Mi,b,W);

% Optimisation function for one first-order and one
% second order transformation function.

S=size(Chrom);

for i=1:S(1);

d=Chrom(i,1);
r=Chrom(i,2);
phi=Chrom(i,3);
p=Chrom(i,4);
psi=Chrom(i,5);

F1=((p^2)-2*p*cos(psi)+1)/((r^2)-2*r*cos(phi)+1);
F2=((r^2)+2*r*cos(phi)+1)/((p^2)+2*p*cos(psi)+1);
F=F1*F2;
c=(F*(1+d)-b*(1-d))/(F*(1+d)+b*(1-d));

k1=((1+d)/(1+c));
k2=((p^2)-2*p*cos(psi)+1)/((r^2)-2*r*cos(phi)+1);
k=k1*k2;

% P = 2Pr+4Pc-1 = 5
P=5;

Mz1=(c*(z.^2)+1)./((z.^2)+d);
Mz2=((z.^4)*(r^2)-2*r*cos(phi)*(z.^2)+1)./((z.^4)-
2*p*cos(psi)*(z.^2)+(p^2));
Mz=k*(z.^P).*Mz1.*Mz2;

f(i,1)=sum(((abs(Mz-Mi)).^2).*W);

end;
```

D1.2 Hybrid optimisation using GA followed by standard methods

```
% tvgaopt3
% GA optimisation of coefficients c,d,r,phi,p,psi
% using the constr.m, fmins.m and the fminu.m functions
% constr.m is a constrained optimisation algorithm using the
% Sequential Quadratic Programming (SQP) method.
% fmins.m is a unconstrained optimisation algorithm using the
% simplex search method of Nelder and Mead.
% fminu.m is a unconstrained optimisation algorithm using the
% default procedure based on BFGS quasi-Newton method. An alternative
% procedure based on DFP formula can be used by setting options(6)=1.

% G.S.Baicher  UWCN
.
.
```

first part of this code is the same as in Appendix 4.1 for 20 generations.

```
.
.
x =[c,d,r,phi,p,psi];
f_ga=f
x_ga=x
xold=x;

% start optimisation algorithm

x(1)=[]; x0=x;

% constrained optimisation  constr.m
x = constr('tvgacob2',x0,[],[],[],[],z,Mi,b,W);

d=x(1); r=x(2); phi=x(3); p=x(4); psi=x(5);

F1=((p^2)-2*p*cos(psi)+1)/((r^2)-2*r*cos(phi)+1);
F2=((r^2)+2*r*cos(phi)+1)/((p^2)+2*p*cos(psi)+1);
F=F1*F2;
c=(F*(1+d)-b*(1-d))/(F*(1+d)+b*(1-d));

k1=((1+d)/(1+c));
k2=((p^2)-2*p*cos(psi)+1)/((r^2)-2*r*cos(phi)+1);
k=k1*k2;

% P = 2Pr+4Pc-1 = 5

P=5;

Mz1=(c*(z.^2)+1)./((z.^2)+d);
Mz2=((z.^4)*(r^2)-2*r*cos(phi)*(z.^2)+1)./((z.^4)-
2*p*cos(psi)*(z.^2)+(p^2));
Mz=k*(z.^P).*Mz1.*Mz2;

f_c3=sum(((abs(Mz-Mi)).^2).*W)
x_c3 = [c,d,r,phi,p,psi]

x=xold;
```

```
x(1)=[]; x0=x;

% unconstrained simplex algorithm fmins.m
x = fmins('tvgasob2',x0,[],[],z,Mi,b,W);

d=x(1); r=x(2); phi=x(3); p=x(4); psi=x(5);

F1=((p^2)-2*p*cos(psi)+1)/((r^2)-2*r*cos(phi)+1);
F2=((r^2)+2*r*cos(phi)+1)/((p^2)+2*p*cos(psi)+1);
F=F1*F2;
c=(F*(1+d)-b*(1-d))/(F*(1+d)+b*(1-d));

k1=((1+d)/(1+c));
k2=((p^2)-2*p*cos(psi)+1)/((r^2)-2*r*cos(phi)+1);
k=k1*k2;

% P = 2Pr+4Pc-1 = 5
P=5;

Mz1=(c*(z.^2)+1)./((z.^2)+d);
Mz2=((z.^4)*(r^2)-2*r*cos(phi)*(z.^2)+1)./((z.^4)-
2*p*cos(psi)*(z.^2)+(p^2));
Mz=k*(z.^P).*Mz1.*Mz2;

f_s3=sum(((abs(Mz-Mi)).^2).*W)
x_s3 = [c,d,r,phi,p,psi]

x=xold;
x(1)=[]; x0=x;

% unconstrained quasi-Newton algorithm fminu.m
x = fminu('tvgasob2',x0,[],[],z,Mi,b,W);

d=x(1); r=x(2); phi=x(3); p=x(4); psi=x(5);

F1=((p^2)-2*p*cos(psi)+1)/((r^2)-2*r*cos(phi)+1);
F2=((r^2)+2*r*cos(phi)+1)/((p^2)+2*p*cos(psi)+1);
F=F1*F2;
c=(F*(1+d)-b*(1-d))/(F*(1+d)+b*(1-d));

k1=((1+d)/(1+c));
k2=((p^2)-2*p*cos(psi)+1)/((r^2)-2*r*cos(phi)+1);
k=k1*k2;

% P = 2Pr+4Pc-1 = 5
P=5;

Mz1=(c*(z.^2)+1)./((z.^2)+d);
Mz2=((z.^4)*(r^2)-2*r*cos(phi)*(z.^2)+1)./((z.^4)-
2*p*cos(psi)*(z.^2)+(p^2));
Mz=k*(z.^P).*Mz1.*Mz2;

f_qN3=sum(((abs(Mz-Mi)).^2).*W)
x_qN3 = [c,d,r,phi,p,psi]
```

```
function f=tvgasob2(x0,z,Mi,b,W);

% Objective function for one first-order and one
% second order transformation function.

%  G.S.Baicher  UWCN

x=x0;
d=x(1); r=x(2); phi=x(3); p=x(4); psi=x(5);

F1=((p^2)-2*p*cos(psi)+1)/((r^2)-2*r*cos(phi)+1);
F2=((r^2)+2*r*cos(phi)+1)/((p^2)+2*p*cos(psi)+1);
F=F1*F2;
c=(F*(1+d)-b*(1-d))/(F*(1+d)+b*(1-d));

k1=((1+d)/(1+c));
k2=((p^2)-2*p*cos(psi)+1)/((r^2)-2*r*cos(phi)+1);
k=k1*k2;

% P = 2Pr+4Pc-1 = 5
P=5;

Mz1=(c*(z.^2)+1)./((z.^2)+d);
Mz2=((z.^4)*(r^2)-2*r*cos(phi)*(z.^2)+1)./((z.^4)-
2*p*cos(psi)*(z.^2)+(p^2));
Mz=k*(z.^P).*Mz1.*Mz2;

f=sum(((abs(Mz-Mi)).^2).*W);
```

D2 Matlab utility codes for deriving QMF bank IIR filters

D2.1 Deriving QMF bank IIR filter transfer functions

```
% transformation of variable method utility (for Pr=Pc=1)
%
% [H0b,H0a,F0b,F0a]=tvfilter(c,d,r,phi,p,psi)
%
% H0b,F0b: numerator of digital filter
% H0a,F0a: denominator of digital filter
% design example 2

c=0.7491;d=0.5001;r=0.1918;phi=1.0318;p=0.1097;psi=1.0013;

k1=((1+d)/(1+c));
k2=((p^2)-2*p*cos(psi)+1)/((r^2)-2*r*cos(phi)+1);
k=k1*k2;

Z1=[k*c(1) 0 k 0 0 0 0 0];
Z2=[1 0 d(1)];
Z3=[r^2 0 -2*r*cos(phi) 0 1];
Z4=[1 0 -2*p*cos(psi) 0 p^2];

Zn=conv(Z1,Z3);
Zd=conv(Z2,Z4);

H1(1:5)=Zn(1:5);
H1(6:12)=Zn(6:12)+Zd;
H2(1:5)=Zn(1:5);
H2(6:12)=Zn(6:12)-3*Zd;
H4=conv(H1,H2);
H5=4*conv(Zd,Zd);

Z2n=conv(Zn,Zn);
Z2d=conv(Zd,Zd);
F2n=conv(Zn,Zd);
F1(1:5)=Zn(1:5);
F1(6:12)=Zn(6:12)+Zd;
F2(1:5)=Z2n(1:5);
F2(6:23)=Z2n(6:23)+F2n;
F2(11:23)=F2(11:23)-8*Z2d;
F3=conv(Zd,Z2d);
F4=conv(F1,F2);
F5=12*F3;

H0b=-H4/4; H0a=H5/4; F0b=-F4/12; F0a=F5/12;
for i=1:(length(F0b)-1);
   H1b(i)=((-1)^(i))*F0b(i);
end;
for i=1:(length(F0a)-1);
   H1a(i)=-1*((-1)^(i))*F0a(i);
end;

H1b(length(F0b))=F0b(length(F0b));
H1a(length(F0a))=F0a(length(F0a));
H1a((length(H1a)+1):(length(H1b)))=0;
```

```
for i=1:(length(H0b)-1);
   F1b(i)=-1*((-1)^(i))*H0b(i);
end;
for i=1:(length(H0a)-1);
   F1a(i)=-1*((-1)^(i))*H0a(i);
end;

F1b(length(H0b))=H0b(length(H0b));
F1a(length(H0a))=H0a(length(H0a));
F1a((length(F1a)+1):(length(F1b)))=0;

H0a((length(H0a)+1):(length(H0b)))=0;
F0a((length(F0a)+1):(length(F0b)))=0;
```

D2.2 Generating IIR filter coefficients and scaling utility

```
function [C,I]=sos2v(sos)

% sos2v.m

% [C,I]=sos2v(sos)

% Creates a vector of coefficients given the sos coefficient matrix
% Matrix I contains the rows of sos having numbers greater than
0.9999695

s=size(sos);
n=[6 5 3 2 1];
m=[-1 -1 1 1 1];
c=1;
I=0;

for i=1:s(1);
   flag=1;
   for r=1:5;
      u=r+(i*5)-5;
      C(u)=sos(i,n(r));
      C(u)=C(u)*m(r);
      if C(u)>0.9999695
            if flag==1
            I(c)=i;
            c=c+1;
            flag=0;
         end;
      elseif C(u)<-1
         if flag==1
            I(c)=i;
            c=c+1;
            flag=0;
         end;
      end;
   end;
end;


function [sos2]=scsos(sos,I,S);

% scsos.m
%
% [sos2]=scsos(sos,I,S);
%
% Scales the b coefficients of sos matrix by the factors contained in
% S. Matrix I contains the rows to be scaled.

l=length(I);
sos2=sos;

for i=1:l;
   sos2(I(i),1:3)=sos2(I(i),1:3)./S(i);
end;
```

Appendix E

GA code for M-Channel uniform filter bank

E1 GA optimisation code for a M-Channel uniform filter bank

E1.1 GA code for optimisation of a M-Channel uniform filter bank

```
% Genetic Algorithm to optimise the amplitude distortion
% and aliasing error of an 8-channel uniform multirate
% filter bank using the square root raised cosine prototype
% filter and cosine modulation method
% GA characteristics
GGAP=0.8;       % generational gap
INSR=0.8;       % reinsertion rate
MAXGEN=10;      % number of generations
Nind=100;       % pop size
MutRate=0.1;    % mutation rate

FieldDR = [13 0 0.1;     % variables: Mp, r and alpha: lower bound
      19 2 0.9];         % upper bound

% create initial population
Chrom = crtrp(Nind, FieldDR);

Best = NaN*ones(MAXGEN+1,1);  % set counter
gen = 0
ObjVal = ufga_obj(Chrom);
[Best(gen+1),ix] = min(ObjVal);
acbest = Chrom(ix,:);
while gen < MAXGEN
   FitnV = ranking(ObjVal);
   SelCh = select('sus',Chrom,FitnV,GGAP);
   SelCh = recombin('recdis', SelCh, 1);
   SelCh = mutbga(SelCh,FieldDR, MutRate);
   ObjVOff = feval('ufga_obj',SelCh);
   [Chrom, ObjVal] = reins(Chrom, SelCh,1,1, ObjVal, ObjVOff);
   gen = gen + 1
   [Best(gen+1,1),ix] = min(ObjVal)
   acbest = Chrom(ix,:);
end

a0=acbest(1); a1=acbest(2); a2=acbest(3);

N = 141;     % number of filter coefficients
Mp= a0;      % prototype filter bandwidth variable
r = a1;      % roll off value 0<r<1
alpha=a2;    % trade-off parameter

% Square root raised-cosine prototype filter

nl=-((N-1)/2):-1;
nu=1:((N-1)/2);
hlsrcn=((4*r*nl).*(cos(((pi*nl*(1+r))/Mp)))+(Mp*sin(((pi*nl*(1-
r))/Mp)));
hlsrcd=(1-((4*r*nl/Mp).*(4*r*nl/Mp))).*(pi*nl*Mp);
hlsrc=hlsrcn./hlsrcd;

husrcn=((4*r*nu).*(cos(((pi*nu*(1+r))/Mp)))+(Mp*sin(((pi*nu*(1-
r))/Mp)));
```

```
husrcd=(1-((4*r*nu/Mp).*(4*r*nu/Mp))).*(pi*nu*Mp);
husrc=husrcn./husrcd;

hscr0=(1/Mp)+((r/Mp)*((4/pi)-1));
hsrc=[hlsrc hscr0 husrc];
[H,w]=freqz(hsrc,1);
plot(w/pi/2,20*log10(abs(H))); grid;

M=8;  % number of channels

% analysis filters

b=hsrc;
n=1:length(b);
b0=(2*b).*cos(((0+(1/2))*((n-((N+1)/2))*pi/M))+((-1)^0)*(pi/4));
[h0,w]=freqz(b0,1,w); h0=abs(h0);
b1=(2*b).*cos(((1+(1/2))*((n-((N+1)/2))*pi/M))+((-1)^1)*(pi/4));
[h1,w]=freqz(b1,1,w); h1=abs(h1);
b2=(2*b).*cos(((2+(1/2))*((n-((N+1)/2))*pi/M))+((-1)^2)*(pi/4));
[h2,w]=freqz(b2,1,w); h2=abs(h2);
b3=(2*b).*cos(((3+(1/2))*((n-((N+1)/2))*pi/M))+((-1)^3)*(pi/4));
[h3,w]=freqz(b3,1,w); h3=abs(h3);
b4=(2*b).*cos(((4+(1/2))*((n-((N+1)/2))*pi/M))+((-1)^4)*(pi/4));
[h4,w]=freqz(b4,1,w); abs(h4);
b5=(2*b).*cos(((5+(1/2))*((n-((N+1)/2))*pi/M))+((-1)^5)*(pi/4));
[h5,w]=freqz(b5,1,w); h5=abs(h5);
b6=(2*b).*cos(((6+(1/2))*((n-((N+1)/2))*pi/M))+((-1)^6)*(pi/4));
[h6,w]=freqz(b6,1,w); h6=abs(h6);
b7=(2*b).*cos(((7+(1/2))*((n-((N+1)/2))*pi/M))+((-1)^7)*(pi/4));
[h7,w]=freqz(b7,1,w); h7=abs(h7);

% Synthesis filters

bs0=(2*b).*cos(((0+(1/2))*((n-((N+1)/2))*pi/M))-((-1)^0)*(pi/4));
[hs0,w]=freqz(bs0,1,w); hs0=abs(hs0);
bs1=(2*b).*cos(((1+(1/2))*((n-((N+1)/2))*pi/M))-((-1)^1)*(pi/4));
[hs1,w]=freqz(bs1,1,w); hs1=abs(hs1);
bs2=(2*b).*cos(((2+(1/2))*((n-((N+1)/2))*pi/M))-((-1)^2)*(pi/4));
[hs2,w]=freqz(bs2,1,w); hs2=abs(hs2);
bs3=(2*b).*cos(((3+(1/2))*((n-((N+1)/2))*pi/M))-((-1)^3)*(pi/4));
[hs3,w]=freqz(bs3,1,w); hs3=abs(hs3);
bs4=(2*b).*cos(((4+(1/2))*((n-((N+1)/2))*pi/M))-((-1)^4)*(pi/4));
[hs4,w]=freqz(bs4,1,w); hs4=abs(hs4);
bs5=(2*b).*cos(((5+(1/2))*((n-((N+1)/2))*pi/M))-((-1)^5)*(pi/4));
[hs5,w]=freqz(bs5,1,w); hs5=abs(hs5);
bs6=(2*b).*cos(((6+(1/2))*((n-((N+1)/2))*pi/M))-((-1)^6)*(pi/4));
[hs6,w]=freqz(bs6,1,w); hs6=abs(hs6);
bs7=(2*b).*cos(((7+(1/2))*((n-((N+1)/2))*pi/M))-((-1)^7)*(pi/4));
[hs7,w]=freqz(bs7,1,w); hs7=abs(hs7);

% amplitude distortion function

T0=conv(b0,bs0); T1=conv(b1,bs1);
T2=conv(b2,bs2); T3=conv(b3,bs3);T4=conv(b4,bs4);
T5=conv(b5,bs5); T6=conv(b6,bs6);T7=conv(b7,bs7);
Tn=(T0+T1+T2+T3+T4+T5+T6+T7);
[T,w]=freqz(Tn,1);
T=abs(T);
```

```
% aliasing fuction

l=1;
e=exp(j*2*pi*l*n/M);

bal0=b0.*e; bal1=b1.*e;bal2=b2.*e; bal3=b3.*e;
bal4=b4.*e; bal5=b5.*e;bal6=b6.*e; bal7=b7.*e;

Ta0=conv(bs0,bal0);Ta1=conv(bs1,bal1);
Ta2=conv(bs2,bal2);Ta3=conv(bs3,bal3);
Ta4=conv(bs4,bal4);Ta5=conv(bs5,bal5);
Ta6=conv(bs6,bal6);Ta7=conv(bs7,bal7);

Tal1=(Ta0+Ta1+Ta2+Ta3+Ta4+Ta5+Ta6+Ta7)/8;
Tal1=abs(Tal1); Tal1=(Tal1.*Tal1);

l=2;
e=exp(j*2*pi*l*n/M);
bal0=b0.*e; bal1=b1.*e;bal2=b2.*e; bal3=b3.*e;
bal4=b4.*e; bal5=b5.*e;bal6=b6.*e; bal7=b7.*e;

Ta0=conv(bs0,bal0);Ta1=conv(bs1,bal1);
Ta2=conv(bs2,bal2);Ta3=conv(bs3,bal3);
Ta4=conv(bs4,bal4);Ta5=conv(bs5,bal5);
Ta6=conv(bs6,bal6);Ta7=conv(bs7,bal7);

Tal2=(Ta0+Ta1+Ta2+Ta3+Ta4+Ta5+Ta6+Ta7)/8;
Tal2=abs(Tal2); Tal2=(Tal2.*Tal2);

l=3;
e=exp(j*2*pi*l*n/M);
bal0=b0.*e; bal1=b1.*e;bal2=b2.*e; bal3=b3.*e;
bal4=b4.*e; bal5=b5.*e;bal6=b6.*e; bal7=b7.*e;

Ta0=conv(bs0,bal0);Ta1=conv(bs1,bal1);
Ta2=conv(bs2,bal2);Ta3=conv(bs3,bal3);
Ta4=conv(bs4,bal4);Ta5=conv(bs5,bal5);
Ta6=conv(bs6,bal6);Ta7=conv(bs7,bal7);

Tal3=(Ta0+Ta1+Ta2+Ta3+Ta4+Ta5+Ta6+Ta7)/8;
Tal3=abs(Tal3); Tal3=(Tal3.*Tal3);
l=4;
e=exp(j*2*pi*l*n/M);
bal0=b0.*e; bal1=b1.*e;bal2=b2.*e; bal3=b3.*e;
bal4=b4.*e; bal5=b5.*e;bal6=b6.*e; bal7=b7.*e;

Ta0=conv(bs0,bal0);Ta1=conv(bs1,bal1);
Ta2=conv(bs2,bal2);Ta3=conv(bs3,bal3);
Ta4=conv(bs4,bal4);Ta5=conv(bs5,bal5);
Ta6=conv(bs6,bal6);Ta7=conv(bs7,bal7);

Tal4=(Ta0+Ta1+Ta2+Ta3+Ta4+Ta5+Ta6+Ta7)/8;
Tal4=abs(Tal4); Tal4=(Tal4.*Tal4);

l=5;
e=exp(j*2*pi*l*n/M);
bal0=b0.*e; bal1=b1.*e;bal2=b2.*e; bal3=b3.*e;
```

```
bal4=b4.*e; bal5=b5.*e;bal6=b6.*e; bal7=b7.*e;

Ta0=conv(bs0,bal0);Ta1=conv(bs1,bal1);
Ta2=conv(bs2,bal2);Ta3=conv(bs3,bal3);
Ta4=conv(bs4,bal4);Ta5=conv(bs5,bal5);
Ta6=conv(bs6,bal6);Ta7=conv(bs7,bal7);

Tal5=(Ta0+Ta1+Ta2+Ta3+Ta4+Ta5+Ta6+Ta7)/8;
Tal5=abs(Tal5); Tal5=(Tal5.*Tal5);

l=6;
e=exp(j*2*pi*l*n/M);
bal0=b0.*e; bal1=b1.*e;bal2=b2.*e; bal3=b3.*e;
bal4=b4.*e; bal5=b5.*e;bal6=b6.*e; bal7=b7.*e;

Ta0=conv(bs0,bal0);Ta1=conv(bs1,bal1);
Ta2=conv(bs2,bal2);Ta3=conv(bs3,bal3);
Ta4=conv(bs4,bal4);Ta5=conv(bs5,bal5);
Ta6=conv(bs6,bal6);Ta7=conv(bs7,bal7);

Tal6=(Ta0+Ta1+Ta2+Ta3+Ta4+Ta5+Ta6+Ta7)/8;
Tal6=abs(Tal6); Tal6=(Tal6.*Tal6);

l=7;
e=exp(j*2*pi*l*n/M);
bal0=b0.*e; bal1=b1.*e;bal2=b2.*e; bal3=b3.*e;
bal4=b4.*e; bal5=b5.*e;bal6=b6.*e; bal7=b7.*e;

Ta0=conv(bs0,bal0);Ta1=conv(bs1,bal1);
Ta2=conv(bs2,bal2);Ta3=conv(bs3,bal3);
Ta4=conv(bs4,bal4);Ta5=conv(bs5,bal5);
Ta6=conv(bs6,bal6);Ta7=conv(bs7,bal7);

Tal7=(Ta0+Ta1+Ta2+Ta3+Ta4+Ta5+Ta6+Ta7)/8;
Tal7=abs(Tal7); Tal7=(Tal7.*Tal7);

Talias=sqrt(Tal1+Tal2+Tal3+Tal4+Tal5+Tal6+Tal7);

% aliasing distortion function
[Tal,w]=freqz(Talias,1);
Tal=abs(Tal);

mbest=a0
rbest=a1
alpha=a2

Epp=max(T)-min(T)
Ea=max(Tal)
err = alpha*(max(T)-min(T))+ (1-alpha)*max(abs(Tal))
```

```
% objective function for 8-ch uniform filter GA
function f = ufga_obj(Chrom);

[Nind,Nvar]=size(Chrom);
for irun = 1:Nind;
   ac = Chrom(irun,:);

a0=ac(1); a1=ac(2); a2=ac(3);

N=141;       % number of filter coefficients
Mp=a0;       % bandwidth variable for the prototype filter
r=a1;        % roll off value
alpha=a2;    % trade-off parameter

---   the following are same as in the main code above ----
% prototype filter
% analysis filters
% synthesis filters
% distortion function
% aliasing fuction
% aliasing error
---          ----         ----         ----
% objective function
err = alpha*(max(T)-min(T))+ (1-alpha)*max(Tal);

f(irun,:) = err;
end
```

Appendix F

GA and hybrid optimisation codes for non-uniform filter banks

F1 GA and hybrid optimisation codes for non-uniform filter banks

F1.1 GA and hybrid optimisation code for non-uniform filter bank using design method - 1

```
% Test for non-uniform filter banks using design method - 1
% with FIR low pass prototype filters with Hamming window as default
% and SGA to optimise the distortion

% GA characteristics

GGAP=0.8;    % gen. gap
INSR=0.8;    % re-insertion rate
MAXGEN=10;   % max. no. of generations
Nind=100;    % pop size
MutRate=1/Nind;    % mutation rate

fc0=(1/4); fc1=(1/24); fc2=(1/12);
fc3=(1/12); fc4=(1/3);
a=0.03;
FieldDR = [-a -a -a -a -a ;    % lower bound
            a a a a a];        % upper bound

Chrom = crtrp(Nind, FieldDR);% create initial population
Best = NaN*ones(MAXGEN+1,1); % counter
gen = 0

ObjVal = nuf5b_obj(Chrom);
[Best(gen+1),ix] = min(ObjVal);
xbest = Chrom(ix,:);

while gen < MAXGEN
   FitnV = ranking(ObjVal);
   SelCh = select('sus',Chrom,FitnV,GGAP);
   SelCh = recombin('recdis', SelCh, 1);
   SelCh = mutbga(SelCh,FieldDR, MutRate);
   ObjVOff = feval('nuf5b_obj',SelCh);
   [Chrom, ObjVal] = reins(Chrom, SelCh,1,1, ObjVal, ObjVOff);
   gen = gen + 1
   [Best(gen+1,1),ix] = min(ObjVal)
   xbest = Chrom(ix,:);
end

x(1)=xbest(1);x(2)=xbest(2);x(3)=xbest(3);x(4)=xbest(4);x(5)=xbest(5);

% second stage for optimisation using Simplex algorithm of the form
% x=fmins('fun5b_obj',x)
wl=512;
M=5;  % number of filter banks
N=17; % odd number giving transition bandwidth
K=M*N;
n=1:K;

% new cut-off frequencies of LP prototype filters
fc0=(1/4)+x(1); fc1=(1/24)+x(2); fc2=(1/12)+x(3);
fc3=(1/12)+x(4); fc4=(1/3)+x(5);
```

```
% generate impulse
in1=zeros(1,9999);
in1=[1 in1];

% transfer function
b0=fir1(K-1,fc0);
b0d=upfirdn(in1,b0,1,4);
b0u=4*upfirdn(b0d,b0,4,1);

b1=fir1(K-1,fc1); wc1=pi*(7/24);
wn1=2*(sin(wc1*(n-((K+1)/2))));
f1na=wn1.*b1; f1ns=-f1na;
b1d=upfirdn(in1,f1na,1,12);
b1u=12*upfirdn(b1d,-f1na,12,1);

b2=fir1(K-1,fc2); wc2=pi*(5/12);
wn2=2*(cos(wc2*(n-((K+1)/2)))); f2na=wn2.*b2;
b2d=upfirdn(in1,f2na,1,6);
b2u=6*upfirdn(b2d,f2na,6,1);

b3=fir1(K-1,fc3); wc3=pi*(7/12);
wn3=2*(sin(wc3*(n-((K+1)/2)))); f3na=wn3.*b3;
b3d=upfirdn(in1,f3na,1,6);
b3u=6*upfirdn(b3d,-f3na,6,1);

b4=fir1(K-1,fc4); wc4=pi;
wn4=(cos(wc4*(n-((K+1)/2)))); f4na=wn4.*b4;
b4d=upfirdn(in1,f4na,1,3);
b4u=3*upfirdn(b4d,f4na,3,1);

bt=b0u(1:10000)+b1u(1:10000)+b2u(1:10000)+b3u(1:10000)+b4u(1:10000);
yt=20*(log10(abs(fft(bt))));
plot(1:5000,yt(1:5000),'b')
axis([0 5000 -2 2]);
xbest

% analysis and synthesis filters

[h0,w]=freqz(b0,1,w); h0=abs(h0); h0=20*log10(h0);
W=w/pi;

wn1=2*(sin(wc1*(n-((K+1)/2))));
f1na=wn1.*b1; f1ns=-f1na;
[h1n,w]=freqz(f1na,1,w); h1n=abs(h1n);
h1n=20*log10(h1n);

wn2=2*(cos(wc2*(n-((K+1)/2)))); f2na=wn2.*b2;
[h2n,w]=freqz(f2na,1,w); h2n=abs(h2n);
h2n=20*log10(h2n);

wn3=2*(sin(wc3*(n-((K+1)/2)))); f3na=wn3.*b3; f3ns=-f3na;
[h3n,w]=freqz(f3na,1,w); h3n=abs(h3n);
h3n=20*log10(h3n);

b4=fir1(K-1,fc4);
wn4=(cos(wc4*(n-((K+1)/2)))); f4na=wn4.*b4;
[h4n,w]=freqz(f4na,1,w); h4n=abs(h4n);
```

```
h4n=20*log10(h4n);
T0=conv(b0,b0); T1=conv(f1na,f1ns);
T2=conv(f2na,f2na); T3=conv(f3na,f3ns);T4=conv(f4na,f4na);
Tn=(T0+T1+T2+T3+T4);

[T,w]=freqz(Tn,1);
T=abs(T);
TdB=20*log10(T);

%aliasing distortion
L=12; r0=4; r1=12; r2=6; r3=6; r4=3;

l=1;
er=exp(j*2*pi*l*n/L);
ba0=b0.*er; Ta10=conv(ba0,b0);
ba1=f1na.*er; Ta11=conv(ba1,f1ns);
ba2=f2na.*er; Ta12=conv(ba2,f2na);
ba3=f3na.*er; Ta13=conv(ba3,f3ns);
ba4=f4na.*er; Ta14=conv(ba4,f4na);

A10=Ta10*(1+exp(-j*2*pi*r0*l/L)+exp(-j*2*pi*r0*2*l/L));
A11=Ta11*1;
A12=Ta12*(1+exp(-j*2*pi*r2*l/L));
A13=Ta13*(1+exp(-j*2*pi*r3*l/L));
A14=Ta14*(1+exp(-j*2*pi*r4*l/L)+exp(-j*2*pi*r4*2*l/L)+exp(-
j*2*pi*r4*3*l/L));

T1=(A10+A11+A12+A13+A14)/L;
T1=abs(T1);T1=T1.*T1;

l=2;
er=exp(j*2*pi*l*n/L);
ba0=b0.*er; Ta10=conv(ba0,b0);
ba1=f1na.*er; Ta11=conv(ba1,f1ns);
ba2=f2na.*er; Ta12=conv(ba2,f2na);
ba3=f3na.*er; Ta13=conv(ba3,f3ns);
ba4=f4na.*er; Ta14=conv(ba4,f4na);
A10=Ta10*(1+exp(-j*2*pi*r0*l/L)+exp(-j*2*pi*r0*2*l/L));
A11=Ta11*1;
A12=Ta12*(1+exp(-j*2*pi*r2*l/L));
A13=Ta13*(1+exp(-j*2*pi*r3*l/L));
A14=Ta14*(1+exp(-j*2*pi*r4*l/L)+exp(-j*2*pi*r4*2*l/L)+exp(-
j*2*pi*r4*3*l/L));
T2=(A10+A11+A12+A13+A14)/L;
T2=abs(T2);T2=T2.*T2;

l=3;
er=exp(j*2*pi*l*n/L);
ba0=b0.*er; Ta10=conv(ba0,b0);
ba1=f1na.*er; Ta11=conv(ba1,f1ns);
ba2=f2na.*er; Ta12=conv(ba2,f2na);
ba3=f3na.*er; Ta13=conv(ba3,f3ns);
ba4=f4na.*er; Ta14=conv(ba4,f4na);
A10=Ta10*(1+exp(-j*2*pi*r0*l/L)+exp(-j*2*pi*r0*2*l/L));
A11=Ta11*1;
A12=Ta12*(1+exp(-j*2*pi*r2*l/L));
A13=Ta13*(1+exp(-j*2*pi*r3*l/L));
```

```
A14=Ta14*(1+exp(-j*2*pi*r4*l/L)+exp(-j*2*pi*r4*2*l/L)+exp(-
j*2*pi*r4*3*l/L));
T3=(A10+A11+A12+A13+A14)/L;
T3=abs(T3);T3=T3.*T3;

l=4;
er=exp(j*2*pi*l*n/L);
ba0=b0.*er; Ta10=conv(ba0,b0);
ba1=f1na.*er; Ta11=conv(ba1,f1ns);
ba2=f2na.*er; Ta12=conv(ba2,f2na);
ba3=f3na.*er; Ta13=conv(ba3,f3ns);
ba4=f4na.*er; Ta14=conv(ba4,f4na);
A10=Ta10*(1+exp(-j*2*pi*r0*l/L)+exp(-j*2*pi*r0*2*l/L));
A11=Ta11*1;
A12=Ta12*(1+exp(-j*2*pi*r2*l/L));
A13=Ta13*(1+exp(-j*2*pi*r3*l/L));
A14=Ta14*(1+exp(-j*2*pi*r4*l/L)+exp(-j*2*pi*r4*2*l/L)+exp(-
j*2*pi*r4*3*l/L));
T4=(A10+A11+A12+A13+A14)/L;
T4=abs(T4);T4=T4.*T4;

l=5;
er=exp(j*2*pi*l*n/L);
ba0=b0.*er; Ta10=conv(ba0,b0);
ba1=f1na.*er; Ta11=conv(ba1,f1ns);
ba2=f2na.*er; Ta12=conv(ba2,f2na);
ba3=f3na.*er; Ta13=conv(ba3,f3ns);
ba4=f4na.*er; Ta14=conv(ba4,f4na);
A10=Ta10*(1+exp(-j*2*pi*r0*l/L)+exp(-j*2*pi*r0*2*l/L));
A11=Ta11*1;
A12=Ta12*(1+exp(-j*2*pi*r2*l/L));
A13=Ta13*(1+exp(-j*2*pi*r3*l/L));
A14=Ta14*(1+exp(-j*2*pi*r4*l/L)+exp(-j*2*pi*r4*2*l/L)+exp(-
j*2*pi*r4*3*l/L));
T5=(A10+A11+A12+A13+A14)/L;
T5=abs(T5);T5=T5.*T5;

l=6;
er=exp(j*2*pi*l*n/L);
ba0=b0.*er; Ta10=conv(ba0,b0);
ba1=f1na.*er; Ta11=conv(ba1,f1ns);
ba2=f2na.*er; Ta12=conv(ba2,f2na);
ba3=f3na.*er; Ta13=conv(ba3,f3ns);
ba4=f4na.*er; Ta14=conv(ba4,f4na);
A10=Ta10*(1+exp(-j*2*pi*r0*l/L)+exp(-j*2*pi*r0*2*l/L));
A11=Ta11*1;
A12=Ta12*(1+exp(-j*2*pi*r2*l/L));
A13=Ta13*(1+exp(-j*2*pi*r3*l/L));
A14=Ta14*(1+exp(-j*2*pi*r4*l/L)+exp(-j*2*pi*r4*2*l/L)+exp(-
j*2*pi*r4*3*l/L));
T6=(A10+A11+A12+A13+A14)/L;
T6=abs(T6);T6=T6.*T6;

l=7;
er=exp(j*2*pi*l*n/L);
ba0=b0.*er; Ta10=conv(ba0,b0);
ba1=f1na.*er; Ta11=conv(ba1,f1ns);
ba2=f2na.*er; Ta12=conv(ba2,f2na);
```

```
ba3=f3na.*er; Ta13=conv(ba3,f3ns);
ba4=f4na.*er; Ta14=conv(ba4,f4na);
A10=Ta10*(1+exp(-j*2*pi*r0*l/L)+exp(-j*2*pi*r0*2*l/L));
A11=Ta11*1;
A12=Ta12*(1+exp(-j*2*pi*r2*l/L));
A13=Ta13*(1+exp(-j*2*pi*r3*l/L));
A14=Ta14*(1+exp(-j*2*pi*r4*l/L)+exp(-j*2*pi*r4*2*l/L)+exp(-
j*2*pi*r4*3*l/L));
T7=(A10+A11+A12+A13+A14)/L;
T7=abs(T7);T7=T7.*T7;

l=8;
er=exp(j*2*pi*l*n/L);
ba0=b0.*er; Ta10=conv(ba0,b0);
ba1=f1na.*er; Ta11=conv(ba1,f1ns);
ba2=f2na.*er; Ta12=conv(ba2,f2na);
ba3=f3na.*er; Ta13=conv(ba3,f3ns);
ba4=f4na.*er; Ta14=conv(ba4,f4na);
A10=Ta10*(1+exp(-j*2*pi*r0*l/L)+exp(-j*2*pi*r0*2*l/L));
A11=Ta11*1;
A12=Ta12*(1+exp(-j*2*pi*r2*l/L));
A13=Ta13*(1+exp(-j*2*pi*r3*l/L));
A14=Ta14*(1+exp(-j*2*pi*r4*l/L)+exp(-j*2*pi*r4*2*l/L)+exp(-
j*2*pi*r4*3*l/L));
T8=(A10+A11+A12+A13+A14)/L;
T8=abs(T8);T8=T8.*T8;

l=9;
er=exp(j*2*pi*l*n/L);
ba0=b0.*er; Ta10=conv(ba0,b0);
ba1=f1na.*er; Ta11=conv(ba1,f1ns);
ba2=f2na.*er; Ta12=conv(ba2,f2na);
ba3=f3na.*er; Ta13=conv(ba3,f3ns);
ba4=f4na.*er; Ta14=conv(ba4,f4na);
A10=Ta10*(1+exp(-j*2*pi*r0*l/L)+exp(-j*2*pi*r0*2*l/L));
A11=Ta11*1;
A12=Ta12*(1+exp(-j*2*pi*r2*l/L));
A13=Ta13*(1+exp(-j*2*pi*r3*l/L));
A14=Ta14*(1+exp(-j*2*pi*r4*l/L)+exp(-j*2*pi*r4*2*l/L)+exp(-
j*2*pi*r4*3*l/L));
T9=(A10+A11+A12+A13+A14)/L;
T9=abs(T9);T9=T9.*T9;

l=10;
er=exp(j*2*pi*l*n/L);
ba0=b0.*er; Ta10=conv(ba0,b0);
ba1=f1na.*er; Ta11=conv(ba1,f1ns);
ba2=f2na.*er; Ta12=conv(ba2,f2na);
ba3=f3na.*er; Ta13=conv(ba3,f3ns);
ba4=f4na.*er; Ta14=conv(ba4,f4na);
A10=Ta10*(1+exp(-j*2*pi*r0*l/L)+exp(-j*2*pi*r0*2*l/L));
A11=Ta11*1;
A12=Ta12*(1+exp(-j*2*pi*r2*l/L));
A13=Ta13*(1+exp(-j*2*pi*r3*l/L));
A14=Ta14*(1+exp(-j*2*pi*r4*l/L)+exp(-j*2*pi*r4*2*l/L)+exp(-
j*2*pi*r4*3*l/L));
T10=(A10+A11+A12+A13+A14)/L;
T10=abs(T10);T10=T10.*T10;
```

```
l=11;
er=exp(j*2*pi*l*n/L);
ba0=b0.*er; Ta10=conv(ba0,b0);
ba1=f1na.*er; Ta11=conv(ba1,f1ns);
ba2=f2na.*er; Ta12=conv(ba2,f2na);
ba3=f3na.*er; Ta13=conv(ba3,f3ns);
ba4=f4na.*er; Ta14=conv(ba4,f4na);
A10=Ta10*(1+exp(-j*2*pi*r0*l/L)+exp(-j*2*pi*r0*2*l/L));
A11=Ta11*1;
A12=Ta12*(1+exp(-j*2*pi*r2*l/L));
A13=Ta13*(1+exp(-j*2*pi*r3*l/L));
A14=Ta14*(1+exp(-j*2*pi*r4*l/L)+exp(-j*2*pi*r4*2*l/L)+exp(-
j*2*pi*r4*3*l/L));
T11=(A10+A11+A12+A13+A14)/L;
T11=abs(T11);T11=T11.*T11;

Talias=sqrt(T1+T2+T3+T4+T5+T6+T7+T8+T9+T10+T11);

[Tal,w]=freqz(Talias,1);
Ea=abs(Tal);
EadB=20*log10(Ea);

Epp=max(T)-min(T);Ea=max(Ea);
fc=[x(1) x(2) x(3) x(4) x(5)];

-------------------------------------------------
% Objective function for the GA code
% NUF bank with 5-bands design example 2
% based on design method -1
function f = nuf5b_obj(Chrom);

M=5;  % number of filter banks
N=17; % odd number giving transition bandwidth
K=M*N;

% starting values

[Nind,Nvar]=size(Chrom);
for irun = 1:Nind;
   x = Chrom(irun,:);

fc0=(1/4)+x(1); fc1=(1/24)+x(2); fc2=(1/12)+x(3);
fc3=(1/12)+x(4); fc4=(1/3)+x(5);
in1=zeros(1,9999);
in1=[1 in1];
in1=in1';

b0=fir1(K-1,fc0);
b0d=upfirdn(in1',b0,1,4);
b0u=4*upfirdn(b0d,b0,4,1);

b1=fir1(K-1,fc1); n=1:K; wc1=pi*(7/24);
wn1=2*(sin(wc1*(n-((K+1)/2))));
f1na=wn1.*b1; f1ns=-f1na;
b1d=upfirdn(in1',f1na,1,12);
b1u=12*upfirdn(b1d,-f1na,12,1);
```

```
b2=firl(K-1,fc2); wc2=pi*(5/12);
wn2=2*(cos(wc2*(n-((K+1)/2)))); f2na=wn2.*b2;
b2d=upfirdn(in1',f2na,1,6);
b2u=6*upfirdn(b2d,f2na,6,1);

b3=firl(K-1,fc3); wc3=pi*(7/12);
wn3=2*(sin(wc3*(n-((K+1)/2)))); f3na=wn3.*b3; f3ns=-f3na;
b3d=upfirdn(in1',f3na,1,6);
b3u=6*upfirdn(b3d,-f3na,6,1);

b4=firl(K-1,fc4); wc4=pi;
wn4=(cos(wc4*(n-((K+1)/2)))); f4na=wn4.*b4;
b4d=upfirdn(in1',f4na,1,3);
b4u=3*upfirdn(b4d,f4na,3,1);

bt=b0u(1:10000)+b1u(1:10000)+b2u(1:10000)+b3u(1:10000)+b4u(1:10000);
yt=20*(log10(abs(fft(bt))));
plot(1:5000,yt(1:5000),'b')
axis([1 5000 -2 2]);
err = max(yt(1:5000))-min(yt(1:5000));
f(irun,:) = err;
end
```

F1.2 GA and hybrid optimisation code for non-uniform filter bank using design method - 2

```
% Test for non-uniform filter banks
% using square root raised cosine prototypes

% GA characteristics

GGAP=0.8;    % gen. gap
INSR=0.8;    % reinsertion rate
MAXGEN=10;   % max. no. of generations
Nind=100;    % pop size
MutRate=1/Nind;

d=0.8;rl=0.5; ru=2.0
FieldDR = [7+d 23+d 11+d 11+d 5+d rl rl rl rl rl;          % lower bound
             9-d 25-d 13-d 13-d 7-d ru ru ru ru ru];       % upper bound

% create initial population
Chrom = crtrp(Nind, FieldDR);
Best = NaN*ones(MAXGEN+1,1);                  % counter
gen = 0

ObjVal = rc5b_obj(Chrom);
[Best(gen+1),ix] = min(ObjVal);
xbest = Chrom(ix,:);

while gen < MAXGEN
   FitnV = ranking(ObjVal);
   SelCh = select('sus',Chrom,FitnV,GGAP);
   SelCh = recombin('recdis', SelCh, 1);
   SelCh = mutbga(SelCh,FieldDR, MutRate);
   ObjVOff = feval('rc5b_obj',SelCh);
   [Chrom, ObjVal] = reins(Chrom, SelCh,1,1, ObjVal, ObjVOff);
   gen = gen + 1
   [Best(gen+1,1),ix] = min(ObjVal)
   xbest = Chrom(ix,:);

end

N=45;                    % number of filter coefficients
M0=4; M1=12; M2=6; M4=3;
x=xbest;

Mp0=x(1); Mp1=x(2); Mp2=x(3); Mp3=x(4); Mp4=x(5);
r0=x(6);r1=x(7);r2=x(8);r3=x(9); r4=x(10);

x=[Mp0,Mp1,Mp2,Mp3,Mp4,r0,r1,r2,r3,r4];

% second stage optimisation using the simplex algorithm of the form
% x=fmins('rc5b_obj',x);

Mp0=x(1); Mp1=x(2); Mp2=x(3); Mp3=x(4); Mp4=x(5);
r0=x(6);r1=x(7);r2=x(8);r3=x(9); r4=x(10);

xbest=[Mp0,Mp1,Mp2,Mp3,Mp4,r0,r1,r2,r3,r4]
```

```
% prototype filter number 0
nl=-((N-1)/2):-1;
nu=1:((N-1)/2);

M=Mp0; r=r0;
hlsrcn=((4*r*nl).*(cos(((pi*nl*(1+r))/M)))+(M*sin(((pi*nl*(1-
r))/M)));
hlsrcd=(1-((4*r*nl/M).*(4*r*nl/M))).*(pi*nl*M);
hlsrc=hlsrcn./hlsrcd;
husrcn=((4*r*nu).*(cos(((pi*nu*(1+r))/M)))+(M*sin(((pi*nu*(1-
r))/M)));
husrcd=(1-((4*r*nu/M).*(4*r*nu/M))).*(pi*nu*M);
husrc=husrcn./husrcd;
hscr0=(1/M)+((r/M)*((4/pi)-1));
hsrc0=[hlsrc hscr0 husrc];

% prototype filter number 1

M=Mp1; r=r1;

hlsrcn=((4*r*nl).*(cos(((pi*nl*(1+r))/M)))+(M*sin(((pi*nl*(1-
r))/M)));
hlsrcd=(1-((4*r*nl/M).*(4*r*nl/M))).*(pi*nl*M);
hlsrc=hlsrcn./hlsrcd;
husrcn=((4*r*nu).*(cos(((pi*nu*(1+r))/M)))+(M*sin(((pi*nu*(1-
r))/M)));
husrcd=(1-((4*r*nu/M).*(4*r*nu/M))).*(pi*nu*M);
husrc=husrcn./husrcd;
hscr0=(1/M)+((r/M)*((4/pi)-1));
hsrc1=[hlsrc hscr0 husrc];

% prototype filter number 2

M=Mp2; r=r2;

hlsrcn=((4*r*nl).*(cos(((pi*nl*(1+r))/M)))+(M*sin(((pi*nl*(1-
r))/M)));
hlsrcd=(1-((4*r*nl/M).*(4*r*nl/M))).*(pi*nl*M);
hlsrc=hlsrcn./hlsrcd;
husrcn=((4*r*nu).*(cos(((pi*nu*(1+r))/M)))+(M*sin(((pi*nu*(1-
r))/M)));
husrcd=(1-((4*r*nu/M).*(4*r*nu/M))).*(pi*nu*M);
husrc=husrcn./husrcd;
hscr0=(1/M)+((r/M)*((4/pi)-1));
hsrc2=[hlsrc hscr0 husrc];

% prototype filter number 3

M=Mp3; r=r3;

hlsrcn=((4*r*nl).*(cos(((pi*nl*(1+r))/M)))+(M*sin(((pi*nl*(1-
r))/M)));
hlsrcd=(1-((4*r*nl/M).*(4*r*nl/M))).*(pi*nl*M);
hlsrc=hlsrcn./hlsrcd;
husrcn=((4*r*nu).*(cos(((pi*nu*(1+r))/M)))+(M*sin(((pi*nu*(1-
r))/M)));
husrcd=(1-((4*r*nu/M).*(4*r*nu/M))).*(pi*nu*M);
```

```
husrc=husrcn./husrcd;
hscr0=(1/M)+((r/M)*((4/pi)-1));
hsrc3=[hlsrc hscr0 husrc];

% prototype filter number 4

M=Mp4; r=r4;

hlsrcn=((4*r*nl).*(cos(((pi*nl*(1+r))/M)))+(M*sin(((pi*nl*(1-
r))/M)));
hlsrcd=(1-((4*r*nl/M).*(4*r*nl/M))).*(pi*nl*M);
hlsrc=hlsrcn./hlsrcd;
husrcn=((4*r*nu).*(cos(((pi*nu*(1+r))/M)))+(M*sin(((pi*nu*(1-
r))/M)));
husrcd=(1-((4*r*nu/M).*(4*r*nu/M))).*(pi*nu*M);
husrc=husrcn./husrcd;
hscr0=(1/M)+((r/M)*((4/pi)-1));
hsrc4=[hlsrc hscr0 husrc];

% analysis and synthesis filters

M=M0;
b=hsrc0;
n=1:length(b);
b0=(2*b).*cos(((0+(1/2))*((n-((N+1)/2))*pi/M))+((-1)^0)*(pi/4));
[h0,w]=freqz(b0,1,w); h0=abs(h0); h0=20*log10(h0);
b0s=(2*b).*cos(((0+(1/2))*((n-((N+1)/2))*pi/M))-((-1)^0)*(pi/4));

M=M1;
b=hsrc1;
b1=(2*b).*cos(((3+(1/2))*((n-((N+1)/2))*pi/M))+((-1)^3)*(pi/4));
[h1,w]=freqz(b1,1,w); h1=20*log10(abs(h1));
b1s=(2*b).*cos(((3+(1/2))*((n-((N+1)/2))*pi/M))-((-1)^3)*(pi/4));

M=M2;
b=hsrc2;
b2=(2*b).*cos(((2+(1/2))*((n-((N+1)/2))*pi/M))+((-1)^2)*(pi/4));
[h2,w]=freqz(b2,1,w); h2=20*log10(abs(h2));
b2s=(2*b).*cos(((2+(1/2))*((n-((N+1)/2))*pi/M))-((-1)^2)*(pi/4));

M=M2;
b=hsrc3;
b3=(2*b).*cos(((3+(1/2))*((n-((N+1)/2))*pi/M))+((-1)^3)*(pi/4));
[h3,w]=freqz(b3,1,w); h3=20*log10(abs(h3));
b3s=(2*b).*cos(((3+(1/2))*((n-((N+1)/2))*pi/M))-((-1)^3)*(pi/4));

M=M4;
b=hsrc4;
b4=(2*b).*cos(((2+(1/2))*((n-((N+1)/2))*pi/M))+((-1)^2)*(pi/4));
[h4,w]=freqz(b4,1,w); h4=20*log10(abs(h4));
b4s=(2*b).*cos(((2+(1/2))*((n-((N+1)/2))*pi/M))-((-1)^2)*(pi/4));

T0=conv(b0,b0s); T1=conv(b1,b1s);
T2=conv(b2,b2s); T3=conv(b3,b3s);T4=conv(b4,b4s);

Tn=(T0+T1+T2+T3+T4);

% distortion function
```

```
[T,w]=freqz(Tn,1);
T=abs(T);
TdB=20*log10(T);

% aliasing fuction

***********  same as in Appendix 6.3.1 **********

[Tal,w]=freqz(Talias,1);
Ea=abs(Tal);
EadB=20*log10(Ea);
Epp=max(T)-min(T)
Ea=max(Ea)

-------------------------------------------------------

% objective function for GA code for 5-band
% NUF bank using design method - 2
function f = rc5b_obj(Chrom);

M0=4; M1=12; M2=6; M4=3;

% starting values

[Nind,Nvar]=size(Chrom);
for irun = 1:Nind;
   x = Chrom(irun,:);
% analysis filters
Mp0=x(1); Mp1=x(2); Mp2=x(3); Mp3=x(4); Mp4=x(5);
r0=x(6);r1=x(7);r2=x(8);r3=x(9); r4=x(10);
N=45;
nl=-((N-1)/2):-1;

M=Mp0; r=r0;
hlsrcn=((4*r*nl).*(cos(((pi*nl*(1+r))/M)))+(M*sin(((pi*nl*(1-
r))/M)));
hlsrcd=(1-((4*r*nl/M).*(4*r*nl/M))).*(pi*nl*M);
hlsrc=hlsrcn./hlsrcd;
husrc=fliplr(hlsrc);
hscr0=(1/M)+((r/M)*((4/pi)-1));
hsrc0=[hlsrc hscr0 husrc];

% prototype filter number 1

M=Mp1; r=r1;

hlsrcn=((4*r*nl).*(cos(((pi*nl*(1+r))/M)))+(M*sin(((pi*nl*(1-
r))/M)));
hlsrcd=(1-((4*r*nl/M).*(4*r*nl/M))).*(pi*nl*M);
hlsrc=hlsrcn./hlsrcd;
husrc=fliplr(hlsrc);
hscr0=(1/M)+((r/M)*((4/pi)-1));
hsrc1=[hlsrc hscr0 husrc];

% prototype filter number 2

M=Mp2; r=r2;
```

```
hlsrcn=((4*r*nl).*(cos(((pi*nl*(1+r))/M)))+(M*sin(((pi*nl*(1-
r))/M)));
hlsrcd=(1-((4*r*nl/M).*(4*r*nl/M))).*(pi*nl*M);
hlsrc=hlsrcn./hlsrcd;
husrc=fliplr(hlsrc);
hscr0=(1/M)+((r/M)*((4/pi)-1));
hsrc2=[hlsrc hscr0 husrc];

% prototype filter number 3

M=Mp3; r=r3;
hlsrcn=((4*r*nl).*(cos(((pi*nl*(1+r))/M)))+(M*sin(((pi*nl*(1-
r))/M)));
hlsrcd=(1-((4*r*nl/M).*(4*r*nl/M))).*(pi*nl*M);
hlsrc=hlsrcn./hlsrcd;
husrc=fliplr(hlsrc);
hscr0=(1/M)+((r/M)*((4/pi)-1));
hsrc3=[hlsrc hscr0 husrc];

% prototype filter number 4
M=Mp4; r=r4;
hlsrcn=((4*r*nl).*(cos(((pi*nl*(1+r))/M)))+(M*sin(((pi*nl*(1-
r))/M)));
hlsrcd=(1-((4*r*nl/M).*(4*r*nl/M))).*(pi*nl*M);
hlsrc=hlsrcn./hlsrcd;
husrc=fliplr(hlsrc);
hscr0=(1/M)+((r/M)*((4/pi)-1));
hsrc4=[hlsrc hscr0 husrc];

% analysis and synthesis filters
M=M0;
b=hsrc0;
n=1:length(b);
b0=(2*b).*cos(((0+(1/2))*((n-((N+1)/2))*pi/M))+((-1)^0)*(pi/4));
b0s=(2*b).*cos(((0+(1/2))*((n-((N+1)/2))*pi/M))-((-1)^0)*(pi/4));
M=M1;
b=hsrc1;
b1=(2*b).*cos(((3+(1/2))*((n-((N+1)/2))*pi/M))+((-1)^3)*(pi/4));
b1s=(2*b).*cos(((3+(1/2))*((n-((N+1)/2))*pi/M))-((-1)^3)*(pi/4));
M=M2;
b=hsrc2;
b2=(2*b).*cos(((2+(1/2))*((n-((N+1)/2))*pi/M))+((-1)^2)*(pi/4));
b2s=(2*b).*cos(((2+(1/2))*((n-((N+1)/2))*pi/M))-((-1)^2)*(pi/4));
M=M2;
b=hsrc3;
b3=(2*b).*cos(((3+(1/2))*((n-((N+1)/2))*pi/M))+((-1)^3)*(pi/4));
b3s=(2*b).*cos(((3+(1/2))*((n-((N+1)/2))*pi/M))-((-1)^3)*(pi/4));
M=M4;
b=hsrc4;
b4=(2*b).*cos(((2+(1/2))*((n-((N+1)/2))*pi/M))+((-1)^2)*(pi/4));
b4s=(2*b).*cos(((2+(1/2))*((n-((N+1)/2))*pi/M))-((-1)^2)*(pi/4));

% transfer function

***********  same as in Appendix 6.3.1 *************
err=max(yt(1:5000))-min(yt(1:5000));
f(irun,:) = err;
end
```

F2 Coefficients for the MELP decoder filter bank

F2.1 Coefficients taken from MELP US Federal Standard – Appendix A

0-500 Hz	500-1000 Hz	1000-2000 Hz	2000-3000 Hz	3000-4000 Hz
-0.00302890	-0.00249420	-0.00231491	0.00231491	0.00554149
-0.00701117	-0.00282091	0.00990113	0.00990113	-0.00981750
-0.01130619	0.00257679	0.02086129	-0.02086129	0.00856805
-0.01494082	0.01451271	0.00000000	0.00000000	0.00000000
-0.01672586	0.02868120	-0.03086123	0.03086123	-0.01267517
-0.01544189	0.03621179	-0.02180695	-0.02180695	0.02162277
-0.01006619	0.02784469	0.00769333	-0.00769333	-0.01841647
0.00000000	0.00000000	0.00000000	0.00000000	0.00000000
0.01474923	-0.04079870	-0.01127245	0.01127245	0.02698425
0.03347158	-0.07849207	0.04726837	0.04726837	-0.04686914
0.05477206	-0.09392213	0.10106105	-0.10106105	0.04150730
0.07670890	-0.07451087	0.00000000	0.00000000	0.00000000
0.09703726	-0.02211575	-0.17904543	0.17904543	-0.07353666
0.11352143	0.04567473	-0.16031428	-0.16031428	0.15896026
0.12426379	0.10232715	0.09497157	-0.09497157	-0.22734513
0.12799355	0.12432599	0.25562154	0.25562154	0.25346255
0.12426379	0.10232715	0.09497157	-0.09497157	-0.22734513
0.11352143	0.04567473	-0.16031428	-0.16031428	0.15896026
0.09703726	-0.02211575	-0.17904543	0.17904543	-0.07353666
0.07670890	-0.07451087	0.00000000	0.00000000	0.00000000
0.05477206	-0.09392213	0.10106105	-0.10106105	0.04150730
0.03347158	-0.07849207	0.04726837	0.04726837	-0.04686914
0.01474923	-0.04079870	-0.01127245	0.01127245	0.02698425
0.00000000	0.00000000	0.00000000	0.00000000	0.00000000
-0.01006619	0.02784469	0.00769333	-0.00769333	-0.01841647
-0.01544189	0.03621179	-0.02180695	-0.02180695	0.02162277
-0.01672586	0.02868120	-0.03086123	0.03086123	-0.01267517
-0.01494082	0.01451271	0.00000000	0.00000000	0.00000000
-0.01130619	0.00257679	0.02086129	-0.02086129	0.00856805
-0.00701117	-0.00282091	0.00990113	0.00990113	-0.00981750
-0.00302890	-0.00249420	-0.00231491	0.00231491	0.00554149

F2.2 Optimised coefficients for the MELP decoder derived using design method – 1.

0-500 Hz	500-1000 Hz	1000-2000 Hz	2000-3000 Hz	3000-4000 Hz
0.00115449	0.00177634	0.00063104	0.00232510	-0.00079102
0.00056857	0.00294002	-0.00016315	-0.00102785	-0.00077181
-0.00054142	0.00316472	0.00208261	0.00024646	0.00279205
-0.00270295	0.00175675	0.00000000	-0.00490030	-0.00378835
-0.00603678	-0.00003817	-0.01191956	0.00446913	0.00097558
-0.00982600	0.00231816	-0.01366229	0.01389190	0.00654825
-0.01235591	0.01344729	0.00862268	-0.02344390	-0.01408511
-0.01117221	0.03117056	0.01804098	0.00000000	0.01227760
-0.00375041	0.04242094	0.00057685	0.00891769	0.00513561
0.01160217	0.02866649	0.02099278	0.01470710	-0.03166337
0.03491244	-0.01975593	0.06936930	0.02594843	0.04589679
0.06409449	-0.09044007	0.00000000	-0.12722144	-0.02243309
0.09510722	-0.14936487	-0.17408769	0.07299819	-0.04911514
0.12274316	-0.15883315	-0.16935148	0.17507684	0.15083362
0.14185831	-0.10258844	0.10508014	-0.26516237	-0.24116031
0.14868968	0.00000000	0.28708773	0.00000000	0.27729118
0.14185831	0.10258844	0.10508014	0.26516237	-0.24116031
0.12274316	0.15883315	-0.16935148	-0.17507684	0.15083362
0.09510722	0.14936487	-0.17408769	-0.07299819	-0.04911514
0.06409449	0.09044007	0.00000000	0.12722144	-0.02243309
0.03491244	0.01975593	0.06936930	-0.02594843	0.04589679
0.01160217	-0.02866649	0.02099278	-0.01470710	-0.03166337
-0.00375041	-0.04242094	0.00057685	-0.00891769	0.00513561
-0.01117221	-0.03117056	0.01804098	0.00000000	0.01227760
-0.01235591	-0.01344729	0.00862268	0.02344390	-0.01408511
-0.00982600	-0.00231816	-0.01366229	-0.01389190	0.00654825
-0.00603678	0.00003817	-0.01191956	-0.00446913	0.00097558
-0.00270295	-0.00175675	0.00000000	0.00490030	-0.00378835
-0.00054142	-0.00316472	0.00208261	-0.00024646	0.00279205
0.00056857	-0.00294002	-0.00016315	0.00102785	-0.00077181
0.00115449	-0.00177634	0.00063104	-0.00232510	-0.00079102